A Proximal Block Coordinate Descent Algorithm for Deep Neural Network Training

Tim Tsz-Kit Lau[†], **Jinshan Zeng**[†], **Baoyuan Wu**[‡], **Yuan Yao**[†]

[†]Department of Mathematics, Hong Kong University of Science and Technology [‡]Tencent AI Lab

Introduction

- Training deep neural networks (DNNs) efficiently is challenging due to the associated **highly nonconvex** optimization
- The backpropagation (**backprop**) algorithm has long been the most widely used algorithm for gradient computation of parameters of DNNs
- Backprop suffers from various issues, e.g., vanishing gradients
- Various methods to alleviate this issue, e.g., ReLUs and LSTM, but unable to completely tackle this inherent problem to backprop
- One viable alternative is to adopt gradient-free methods, including (but not limited to) the alternating direction method of multipliers (**ADMM**) and the block coordinate descent (**BCD**)
- Mainly to decompose the highly coupled and composite DNN training objective into several *loosely coupled* and *almost separable* simple subproblems

Contributions

- Propose a novel algorithm based on **BCD of Gauss-Seidel type**
- Define the loss function using the quadratic penalty method by unrolling the nested structure of DNNs into separate "single-layer" training tasks
- Simplifications of commonly used activation functions as **projections** onto *nonempty closed convex sets*
- The overall loss function is **block multiconvex**
- Obtain global convergence guarantees under the framework of the **Kurdyka-Łojasiewicz (KL) property**

Problem Formulation

- **Notations:**
  - A feedforward neural network with L hidden layers
  - d_ℓ: the number of nodes of the ℓ-th layer
  - N: the number of training samples
  - K: the number of classes, i.e., $d_{L+1} = K$
  - $\boldsymbol{x}_j \in \mathbb{R}^{d_0}$: the j-th training data, $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) \in \mathbb{R}^{d_0 \times N}$
  - $\boldsymbol{y}_j \in \mathbb{R}^{d_{L+1}}$: the one-hot vector of its corresponding label, y_{ji}: the i-th entry of the column vector $\boldsymbol{y}_j$, $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) \in \mathbb{R}^{d_{L+1} \times N}$
  - $\boldsymbol{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$: the weight matrix between the ℓ-th and $(\ell-1)$-th hidden layers
  - $\boldsymbol{b}_\ell \in \mathbb{R}^{d_\ell}$: the bias vector of the ℓ-th hidden layer
  - $\boldsymbol{W}_{L+1} \in \mathbb{R}^{d_{L+1} \times d_L}$: the weight matrix between the last hidden layer and the output layer
  - $\boldsymbol{b}_{L+1} \in \mathbb{R}^{L+1}$: the bias vector of the output layer
  - h: general activation function
  - $\mathcal{W} := \{\boldsymbol{W}_\ell\}_{\ell=1}^{L+1}$ and $\boldsymbol{b} := \{\boldsymbol{b}_\ell\}_{\ell=1}^{L+1}$
- Optimization problem of regularized DNNs:

$$\min_{\boldsymbol{a},\boldsymbol{z},\mathcal{W},\boldsymbol{b}} F(\boldsymbol{a},\mathcal{W},\boldsymbol{b}) \equiv \gamma_{L+1} \sum_{j=1}^{N} \mathcal{L}(\boldsymbol{W}_{L+1}\boldsymbol{a}_{L,j} + \boldsymbol{b}_{L+1}; \boldsymbol{y}_j) + \sum_{\ell=1}^{L+1} r_\ell(\boldsymbol{W}_\ell)$$
$$\text{subject to } \boldsymbol{z}_{\ell,j} = \boldsymbol{W}_\ell \boldsymbol{a}_{\ell-1,j} + \boldsymbol{b}_\ell,\ \boldsymbol{a}_{\ell,j} = h(\boldsymbol{z}_{\ell,j}) \text{ for all } \ell \in \{1,\ldots,L\}, j \in \{1,\ldots,N\} \tag{1}$$

  - $\mathcal{L}$: generic loss function
  - r_ℓ: convex but possibly nonsmooth regularizers
  - $\boldsymbol{a} := \{\boldsymbol{a}_{\ell,j}\}_{\ell=1}^{L}$: the set of all activation vectors
  - $\boldsymbol{z} := \{\boldsymbol{z}_{\ell,j}\}_{\ell=1}^{L}$, $\boldsymbol{a}_{0,j} := \boldsymbol{x}_j$ for all $j \in \{1,\ldots,N\}$, and $\gamma_{L+1} > 0$
- Reformulation (**quadratic penalty method**):

$$\min_{\boldsymbol{a},\boldsymbol{z},\mathcal{W},\boldsymbol{b}} F(\boldsymbol{a},\mathcal{W},\boldsymbol{b}) + \sum_{j=1}^{N}\sum_{\ell=1}^{L} \frac{\rho_\ell}{2}\|h(\boldsymbol{z}_{\ell,j}) - \boldsymbol{a}_{\ell,j}\|^2 + \sum_{j=1}^{N}\sum_{\ell=1}^{L} \frac{\gamma_\ell}{2}\|\boldsymbol{W}_\ell \boldsymbol{a}_{\ell-1,j} + \boldsymbol{b}_\ell - \boldsymbol{z}_\ell\|^2 \tag{2}$$

  - $\|\cdot\|$ is the standard Euclidean norm, $\rho_\ell > 0$ and $\gamma_\ell > 0$ for all $\ell \in \{1,\ldots,L\}$
  - Allows for any general activation functions such as hyperbolic tangent, sigmoid and ReLU
  - The formulation (2) is generally hard to solve explicitly
  - Can be simplified if we consider the constraint $\boldsymbol{a}_{\ell,j} = h(\boldsymbol{z}_{\ell,j})$ as a **projection** onto a *convex set*
  - E.g., ReLU can be thought of as a projection onto the closed upper half-space
- Introduction of a set of auxiliary variable $\{\boldsymbol{u}_{\ell,j}\}_{\ell=1}^{L}$:

$$\min_{\boldsymbol{a},\mathcal{W},\boldsymbol{b},\boldsymbol{u}} \widetilde{F}(\boldsymbol{a},\mathcal{W},\boldsymbol{b},\boldsymbol{u}) \equiv F(\boldsymbol{a},\mathcal{W},\boldsymbol{b}) + \sum_{j=1}^{N}\sum_{\ell=1}^{L}\left[\frac{\gamma_\ell}{2}\|\boldsymbol{W}_\ell \boldsymbol{a}_{\ell-1,j} + \boldsymbol{b}_\ell - \boldsymbol{a}_{\ell,j} + \boldsymbol{u}_{\ell,j}\|^2 + \iota_{\mathcal{S}_\ell}(\boldsymbol{a}_{\ell,j})\right] \tag{3}$$

  - $\mathcal{S}_\ell$: a nonempty closed convex set
    - $\iota_{\mathcal{S}_\ell}$: the indicator function of a nonempty closed convex set $\mathcal{S}_\ell$ such that

$$\iota_{\mathcal{S}_\ell}(\boldsymbol{u}) = \begin{cases} 0 & \text{for } \boldsymbol{u} \in \mathcal{S}_\ell \\ +\infty & \text{otherwise} \end{cases}$$

  - For the case of the ReLU activation function, $\mathcal{S}_\ell = \mathbb{R}^{d_\ell}_+$
- The objective function (3) is **block multiconvex** which allows for established convergence guarantees in existing literature using the proposed algorithm

The Proximal BCD Algorithm

- **Input**: training data $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^{N}$ and regularization parameters $\{\gamma_\ell\}_{\ell=1}^{L+1}$, $\boldsymbol{a}_{L+1,j}^{(k)} := \boldsymbol{y}_j$ for all $k \in \mathbb{N}$ and $j \in \{1,\ldots,N\}$
- Initialize $\mathcal{W}^{(0)}$, $\boldsymbol{b}^{(0)}$, $\alpha_i^{(0)} \in (0,\infty)$, $\beta_i^{(0)} \in (0,1)$ for all i, and $s,t \in (0,1]$
- $\boldsymbol{a}^{(0)}$ initialized by *forward propagation*
- Iterative update each block using a *proximal step*
- *Adaptive momentums* used in implementations
- Update of the parameters of the last layer:

$$\boldsymbol{W}_{L+1}^{*} = \operatorname*{argmin}_{\boldsymbol{W}_{L+1}} \gamma_{L+1}\mathcal{L}(\boldsymbol{W}_{L+1}\boldsymbol{a}_{L,j}^{(k-1)} + \boldsymbol{b}_{L+1}^{(k-1)}; \boldsymbol{y}_j) + \frac{\alpha_1^{(k-1)}}{2}\|\boldsymbol{W}_{L+1} - \boldsymbol{W}_{L+1}^{(k-1)}\|_F^2 + r_{L+1}(\boldsymbol{W}_{L+1})$$
$$\boldsymbol{W}_{L+1}^{(k)} = \boldsymbol{W}_{L+1}^{(k-1)} + \beta_1^{(k-1)}(\boldsymbol{W}_{L+1}^{*} - \boldsymbol{W}_{L+1}^{(k-1)})$$
$$\boldsymbol{b}_{L+1}^{*} = \operatorname*{argmin}_{\boldsymbol{b}_{L+1}} \gamma_{L+1}\mathcal{L}(\boldsymbol{W}_{L+1}^{(k-1)}\boldsymbol{a}_{L,j}^{(k-1)} + \boldsymbol{b}_{L+1}; \boldsymbol{y}_j) + \frac{\alpha_1^{(k-1)}}{2}\|\boldsymbol{b}_{L+1} - \boldsymbol{b}_{L+1}^{(k-1)}\|^2$$
$$\boldsymbol{b}_{L+1}^{(k)} = \boldsymbol{b}_{L+1}^{(k-1)} + \beta_1^{(k-1)}(\boldsymbol{b}_{L+1}^{*} - \boldsymbol{b}_{L+1}^{(k-1)})$$

- Update of the hidden layers, $\ell = L,\ldots,1$:
  1. **Activation vectors:**

$$\boldsymbol{a}_{\ell,j}^{*} = \operatorname*{argmin}_{\boldsymbol{a}_{\ell,j}} \frac{\gamma_{\ell+1}}{2}\|\boldsymbol{W}_{\ell+1}^{(k)}\boldsymbol{a}_{\ell,j} + \boldsymbol{b}_{\ell+1}^{(k)} - \boldsymbol{a}_{\ell+1,j}^{(k)} + \boldsymbol{u}_{\ell+1,j}^{(k)}\|^2 + \frac{\gamma_\ell}{2}\|\boldsymbol{W}_\ell^{(k-1)}\boldsymbol{a}_{\ell-1,j}^{(k-1)} + \boldsymbol{b}_\ell^{(k-1)} - \boldsymbol{a}_{\ell,j} +$$
$$\boldsymbol{u}_{\ell,j}^{(k)}\|^2 + \frac{\alpha_{2(L-\ell+1)}^{(k-1)}}{2}\|\boldsymbol{a}_{\ell,j} - \boldsymbol{a}_{\ell,j}^{(k-1)}\|^2 + \iota_{\mathcal{S}_\ell}(\boldsymbol{a}_{\ell,j}) \text{ for all } j \in \{1,\ldots,N\}$$
$$\boldsymbol{a}_{\ell,j}^{(k)} = \boldsymbol{a}_{\ell,j}^{(k-1)} + \beta_{2(L-\ell+1)}^{(k-1)}(\boldsymbol{a}_{\ell,j}^{*} - \boldsymbol{a}_{\ell,j}^{(k-1)}) \text{ for all } j \in \{1,\ldots,N\}$$

  2. **Auxiliary variables:**

$$\boldsymbol{u}_{\ell,j}^{(k)} = \operatorname*{argmin}_{\boldsymbol{u}_{\ell,j}} \frac{\gamma_\ell}{2}\|\boldsymbol{W}_\ell^{(k-1)}\boldsymbol{a}_{\ell-1,j}^{(k-1)} + \boldsymbol{b}_\ell^{(k-1)} - \boldsymbol{a}_{\ell,j}^{(k)} + \boldsymbol{u}_{\ell,j}\|^2$$

  3. **Parameters:**

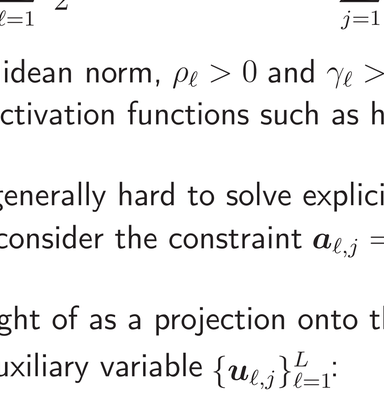
$$\boldsymbol{W}_\ell^{*} = \operatorname*{argmin}_{\boldsymbol{W}_\ell} \sum_{j=1}^{N} \frac{\gamma_\ell}{2}\|\boldsymbol{W}_\ell\boldsymbol{a}_{\ell-1,j}^{(k-1)} + \boldsymbol{b}_\ell^{(k-1)} - \boldsymbol{a}_{\ell,j}^{(k)} + \boldsymbol{u}_{\ell,j}^{(k)}\|^2 + \frac{\alpha_{2(L-\ell+1)+1}^{(k-1)}}{2}\|\boldsymbol{W}_\ell - \boldsymbol{W}_\ell^{(k-1)}\|_F^2 + r_\ell(\boldsymbol{W}_\ell)$$
$$\boldsymbol{W}_\ell^{(k)} = \boldsymbol{W}_\ell^{(k-1)} + \beta_{2(L-\ell+1)+1}^{(k)}(\boldsymbol{W}_\ell^{*} - \boldsymbol{W}_\ell^{(k-1)})$$
$$\boldsymbol{b}_\ell^{*} = \operatorname*{argmin}_{\boldsymbol{b}_\ell} \sum_{j=1}^{N} \frac{\gamma_\ell}{2}\|\boldsymbol{W}_\ell^{(k-1)}\boldsymbol{a}_{\ell-1,j}^{(k-1)} + \boldsymbol{b}_\ell - \boldsymbol{a}_{\ell,j}^{(k)} + \boldsymbol{u}_{\ell,j}^{(k)}\|^2 + \frac{\alpha_{2(L-\ell+1)+1}^{(k-1)}}{2}\|\boldsymbol{b}_\ell - \boldsymbol{b}_\ell^{(k-1)}\|^2$$
$$\boldsymbol{b}_\ell^{(k)} = \boldsymbol{b}_\ell^{(k-1)} + \beta_{2(L-\ell+1)+1}^{(k-1)}(\boldsymbol{b}_\ell^{*} - \boldsymbol{b}_\ell^{(k-1)})$$

Convergence Results

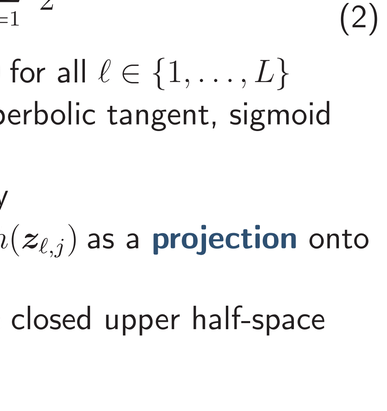
Theorem 1 (Global convergence) Under certain assumptions and the fact that the sequence generated by the proposed algorithm has a finite limit point where $\widetilde{F}$ satisfies the **Kurdyka-Łojasiewicz (KL) property**, *this sequence converges to such a limit point*, which is a critical point of (3).

Experimental Results

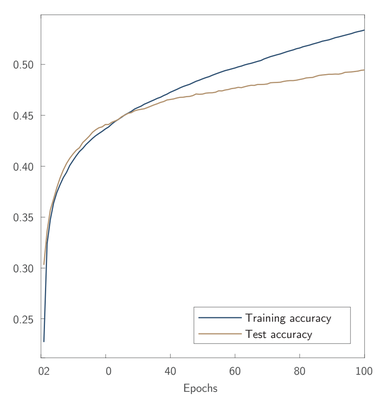
- CIFAR-10 with 50,000 training and 10,000 test samples
- 3,072-4,000-4,000-4,000-10 MLP
- 3,072-4,000-3,072-4,000-10 DNN with a residual connection in the second hidden layer (ResNet)
- The proximal BCD algorithm (20 epochs) and Backprop (SGD; 100 epochs)
- Squared losses; ReLU activations; no regularizations
- Final test accuracies (medians of 5 runs):
  1. MLP with SGD: 0.4765
  2. MLP with BCD: 0.4682
  3. ResNet with SGD: 0.494
  4. ResNet with BCD: 0.4843



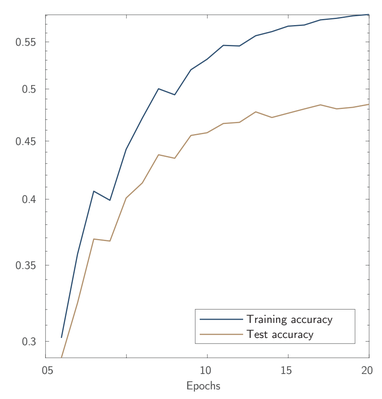
(a) MLP; SGD



(b) MLP; BCD



(c) ResNet; SGD



(d) ResNet; BCD

Conclusion

- Proposed an efficient BCD algorithm and established its convergence guarantees according to our block multiconvex formulation
- Three major advantages of BCD include:
  1. **High per epoch efficiency at early stages** (observed in the figures), i.e., the training and test accuracies of BCD grow much faster than SGD in terms of epoch at the early stage
  2. **Good scalability**, i.e., BCD can be implemented in a distributed and parallel manner via data parallelism on multi-core CPUs
  3. **Gradient free**, i.e., gradient computations are unnecessary used for the updates
- They generally require more memory than SGD method
- A future direction is to study *the feasibility of the stochastic and parallel block coordinate descent methods*