Block Coordinate Descent for Neural Networks Provably Finds Global Minima

Shunta Akiyama
CyberAgent

Neurips 2025

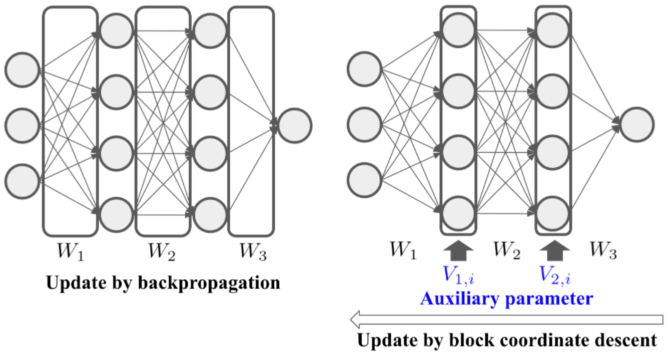
Two-splitting formulation of neural network training

W_j : weight matrix of the j-th layer

$V_{j,i}$: auxiliary variables approximating outputs of the j-th layer

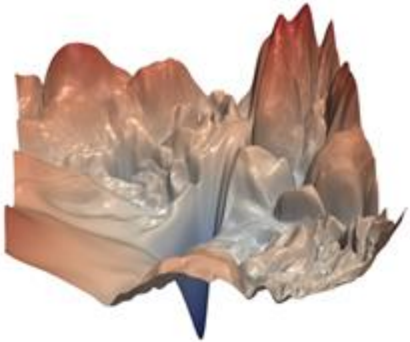
$$\min_{\mathbf{W},\mathbf{V}} \sum_{i=1}^{n} (\underbrace{W_L V_{L-1,i}}_{\text{approximated outputs}} - y_i)^2 \quad \text{s.t.} \quad V_{j,i} = \sigma(W_j V_{j-1,i})$$

$$\min_{\mathbf{W},\mathbf{V}} \sum_{i=1}^{n} (W_L V_{L-1,i} - y_i)^2 + \gamma \sum_{j=1}^{L-1} \sum_{i=1}^{n} \left\| \sigma(W_j V_{j-1,i}) - V_{j,i} \right\|^2$$



Advantage

- Easier to implement in distributed/parallelized manners.
- Decomposition may lead to a preferable loss landscape, while the loss function appearing in deep learning is <u>highly non-convex</u>.

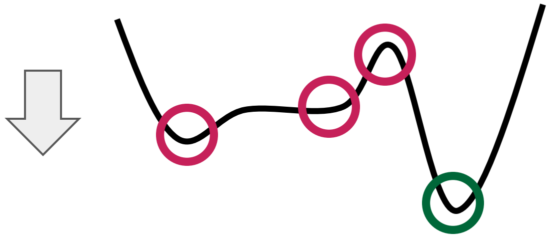


Li, Hao, et al. "Visualizing the loss landscape of neural nets." Neurips 2018.

Our contribution

Existing work Convergence guarantee to **stationary points**

Zhang, Ziming, and Matthew Brand. "Convergent block coordinate descent for training tikhonov regularized deep neural networks." *Neurips 2017*.
Zeng, Jinshan, et al. "Global convergence of block coordinate descent in deep learning." *ICML 2019*.
Lau, Tim Tsz-Kit, et al. "A proximal block coordinate descent algorithm for deep neural network training." *ICLR workshop 2018.*



Stationary points are not always **global minima**

Our results

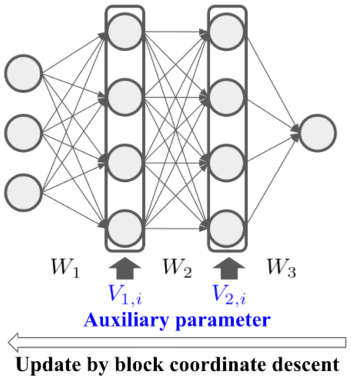
- Convergence guarantee to **global minima**
- Evaluation of **iteration complexity**
- Providing **generalization error bound**

From a viewpoint of DL theory, BCD provides richer theoretical information than backpropagation.

Algorithm

We update the parameters in the backword order:

$$W_L \to V_{L-1,i} \to W_{L-1} \to \cdots \to V_{1,i} \to W_1$$



Algorithm 1: Block Coordinate Descent

Input : K: outer iterations, K_V: inner iterations for $V_{j,i}$, K_W: inner iterations for W_1, η_V: step size for $V_{j,i}$, $\eta_W^{(1)}$, $\eta_W^{(2)}$: step sizes for weight updates

Initialization : $(W_1)_{ab} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, d_{\text{in}}^{-1})$, $(W_j)_{ab} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, r^{-1})$ for $j = 2, \ldots, L$.
Apply singular value bounding to W_j for $j = 2, \ldots, L$ (see Algorithm 2).
Set $V_{0,i} \leftarrow x_i$, and $V_{j,i} \leftarrow \sigma(W_j V_{j-1,i})$ for $j = 1, \ldots, L-1$.

initialization

- **for** $k \leftarrow 1$ **to** K **do**
  - $W_L \leftarrow W_L - \eta_W^{(1)} \nabla_{W_L} \sum_{i=1}^n \|WV_{L-1,i} - y_i\|^2$;
  - **for** $i \leftarrow 1$ **to** n **do**
    - $V_{L-1,i} \leftarrow V_{L-1,i} - \eta_V \nabla_{V_{L-1,i}} \|WV_{L-1,i} - y_i\|^2$;
  - **for** $j \leftarrow L-1$ **to** 2 **do**
    - $W_j \leftarrow W_j - \gamma \eta_W^{(1)} \nabla_{W_j} \sum_{i=1}^n \|\sigma(W_j V_{j-1,i}) - V_{j,i}\|^2$;
    - **for** $i \leftarrow 1$ **to** n **do**
      - **for** $k_{in} \leftarrow 1$ **to** K_V **do**
        - $V_{j-1,i} \leftarrow V_{j-1,i} - \gamma \eta_V \nabla_{V_{j-1,i}} \|\sigma(W_j V_{j-1,i}) - V_{j,i}\|^2$;
  - **for** $k_{in} \leftarrow 1$ **to** K_W **do**
    - $W_1 \leftarrow W_1 - \gamma \eta_W^{(2)} \nabla_{W_1} \sum_{i=1}^n \|\sigma(W_1 V_{0,i}) - V_{1,i}\|^2$;

update (gradient descent)

<u>**Initialization**</u> we apply the **singular value bounding** to each weight matrix.

computing SVD $W_j = U\Sigma V$ ⇨ clipping the singular values in Σ to $[s_1, s_2]$ ⇨ reconstructing $W_j \leftarrow U\Sigma' V$

<u>**Update**</u> we use the gradient descent, e.g.,

$$V_{j-1} \leftarrow V_{j-1,i} - \eta_V \nabla_{V_{L-1,i}} \left\|\sigma(W_j V_{j-1,i}) - V_{j,i}\right\|^2, \qquad W_j \leftarrow W_j - \gamma \eta_W^{(1)} \sum_{i=1}^{n} \nabla_{W_j} \left\|\sigma(W_j V_{j-1,i}) - V_{j,i}\right\|^2$$

Main result : Global convergence guarantee

Assumption

- The activation σ is ℓ-Lipschitz, and its (sub)differential is lower bounded by α.
- The data matrix $X = (x_1, \dots, x_n)^\top \in \mathbb{R}^{n \times d_{in}}$ is row full-rank: $\mathrm{rank}(X) = n$

Theorem Under the setting $\eta_V \leq \frac{\alpha}{16\gamma\ell^4}, \eta_W^{(1)} \leq \frac{\eta_V^{-1}}{8\sqrt{r}C_V K}\left(\frac{\alpha}{2}\right)^L, \eta_W^{(2)} \leq \frac{1}{\gamma\ell^4 \max_i \|x_i\|^2}$, and

$$K = \left\lceil \frac{2}{\eta_V} \log\left(\frac{3R}{\epsilon}\right) \right\rceil,\ K_V = \left\lceil \frac{1}{\gamma\alpha\ell\eta_V} \log\left(\frac{48\gamma\ell^2(L-2)rnC_K^2}{\alpha^2\epsilon}\right) \right\rceil, K_W = \left\lceil \frac{1}{4\gamma s\alpha^2\eta_W^{(2)}} \log\left(\frac{3\ell^2 \max\|x_i\|^2 rnC_K^2}{\alpha^2 s^2 \epsilon}\right) \right\rceil,$$

the output of BCD satisfies

$$F(\mathbf{W}, \mathbf{V}) \leq \epsilon.$$

- The total number of gradient computations is bounded by $O\big(K(LnK_V + K_W)\big) = \tilde{O}(nL \log^2 \epsilon^{-1})$
- We admit **any number of layers**.
- Contrary to the NTK/MF regime, **we do not require any overparameterization**.
- We also provide the **generalization error bound** via Rademacher complexity.

Extension

The same convergence guarantee can be applied to:

- convex loss function
- different activation between layers
- training loss with regularization terms
- multi-dimensional outputs
- **ReLU activation**

Difficulty of ReLU: its only takes **non-negative value**

We need to prevent $V_{j,i}$ from taking negative value due to this non-negativity.

→ **skip connection + non-negative projection**

Theorem

The output of BCD (skip-connection+non-negative projection) applied to ReLU NN satisfies

$$F(\mathbf{W}, \mathbf{V}) \leq \epsilon.$$

$$\min_{\mathbf{W},\mathbf{V}} \sum_{i=1}^{n} (W_L V_{L-1,i} - y_i)^2 \quad \text{s.t.} \quad V_{j,i} = \sigma(W_j V_{j-1,i})$$

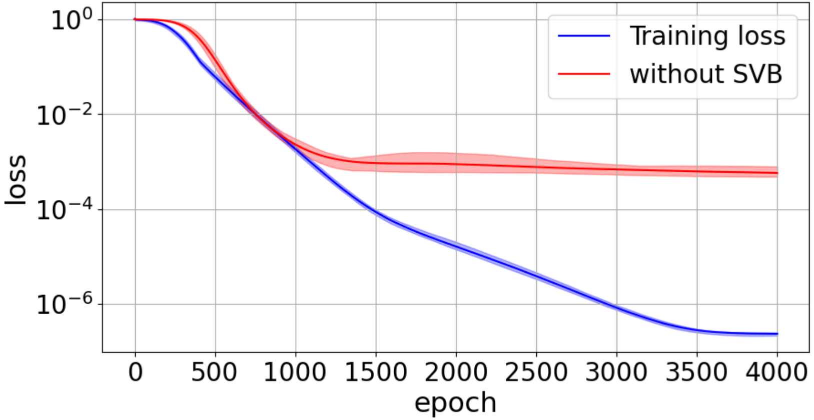
Algorithm 3: Block Coordinate Descent for ReLU Activation

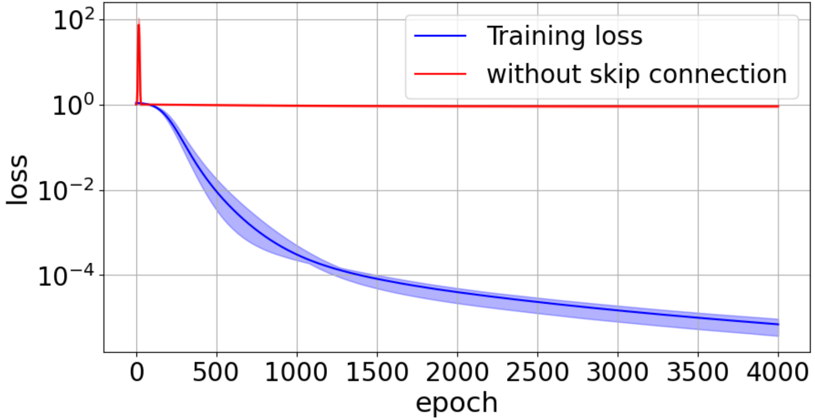
Input : K: outer iterations, K_V: inner iterations for $V_{j,i}$, K_W: inner iterations for W_1, η_V: step size for $V_{j,i}$, $\eta_W^{(1)}, \eta_W^{(2)}$: step sizes for weight updates

Initialization : $(W_1)_{ab} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, d_{\text{in}}^{-1})$, $(W_j)_{ab} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, r^{-1})$ for $j = 2, \dots, L$.
Apply singular value bounding to W_j for $j = 2, \dots, L$ (see Algorithm 2).
Set $V_{0,i} \leftarrow x_i$,
Set $V_{1,i} \leftarrow \sigma(W_1 V_{0,i})$.
Set $V_{j,i} \leftarrow \sigma(W_j V_{j-1,i}) + V_{j-1,i}$ for $j = 2, \dots, L-1$.

```
for k ← 1 to K do
    for i ← 1 to n do
        V_{L-1,i} ← V_{L-1,i} − η_V ∇_{V_{L-1,i}} ||W_L V_{L-1,i} − y_i||^2;
        V_{L-1,i} ← (V_{L-1,i})^+;
    for j ← L − 1 to 2 do
        W_j ← W_j − γ η_W^{(1)} ∇_{W_j} Σ_{i=1}^n ||σ(W_j V_{j-1,i}) + V_{j-1,i} − V_{j,i}||^2;
        for i ← 1 to n do
            for k_in ← 1 to K_V do
                V_{j-1,i} ← V_{j-1,i} − γ η_V ∇_{V_{j-1,i}} ||σ(W_j V_{j-1,i}) + V_{j-1,i} − V_{j,i}||^2;
                V_{j-1,i} ← (V_{j-1,i})^+;
    for k_in ← 1 to K_W do
        W_1 ← W_1 − γ η_W^{(2)} ∇_{W_1} Σ_{i=1}^n ||W_1 V_{0,i} − V_{1,i}||^2;
```

Numerical experiments





- Left: LeakyReLU (negative slope=0.5), Right: ReLU
- Training loss monotonically decreases while the loss of the hidden layers remains small.
- **Skip connection** substantially improves the convergence.