Power of Alternating Direction Method of Multipliers (ADMM) in Deep Learning

Junxiang Wang
Joint works with Zheng Chai (GMU), Yue Cheng (GMU), and Liang Zhao (Emory)

Department of Computer Science and Informatics, Emory University

MOPTA 2021

Table of Contents

1. Background Introduction

2. The dlADMM Algorithm

3. The pdADMM Algorithm

Table of Contents

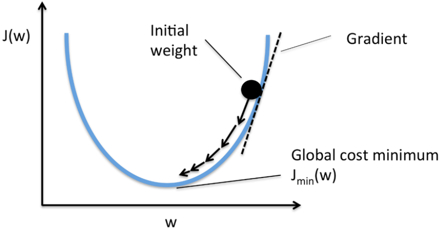
1. Background Introduction

2. The dlADMM Algorithm

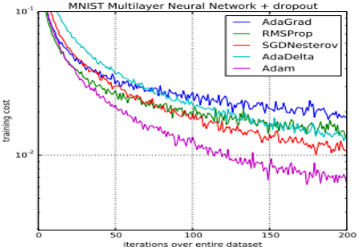
3. The pdADMM Algorithm

SGD as a Deep Learning Optimizer

Stochastic gradient descent(SGD) and its variants are state-of-the-art optimizers in deep learning applications.



Stochastic Gradient Descent(SGD)

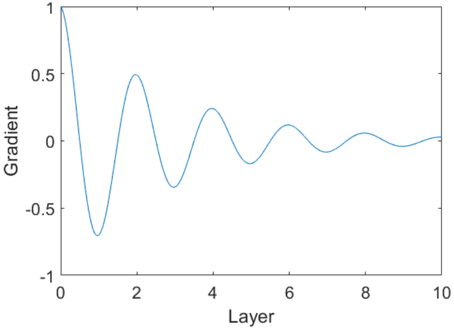


Outstanding Performance

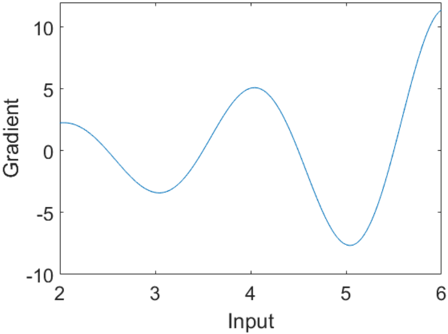
Challenges of SGD

However, SGD suffers from several limitations including:

- Gradient vanishing: the error signal diminishes as the gradient is backpropagated.
- Poor conditioning: small input can change the gradient drastically.



Gradient Vanishing



Poor Conditioning

The Motivations to Train Neural Networks via ADMM

The ADMM has the following advantages over SGD:

- **Be immune to gradient vanishing and poor conditioning.** The ADMM splits a neural network into layerwise components, and prevents the accumulative calculation of gradients.
- **Have great potential of parallelism of deep neural network training.** The inherent nature of ADMM is to break an objective into multiple subproblems, each of which can be solved in parallel; while the parallel training of SGD is restricted by the backward locking (i.e. the calculation of gradient in one layer is dependent on its previous layers).

Existing Works on Training Neural Network using ADMM

Taylor et al. [1] deals with the following Multi-Layer Perceptron (MLP) model:

Problem (MLP Training Problem)

$$\min_{W_l,b_l,z_l,a_l} R(z_L;y)+\sum\nolimits_{l=1}^{L}\Omega_l(W_l)$$
$$s.t.\ z_l=W_la_{l-1}+b_l(l=1,\cdots,L),\ \ a_l=f_l(z_l)(l=1,\cdots,L-1)$$

where $W_l\in\mathbb{R}^{n_l\times n_{l-1}}$ and $b_l\in\mathbb{R}^{n_l}$ are a weight matrix and a bias for the l-th layer, respectively. n_l is the number of neurons for the l-th layer. z_l and a_l are the output of the linear mapping and the nonlinear mapping f_l for the l-th layer, respectively. $x=a_0$ and y are an input matrix and a predefined label vector, respectively. $R(z_L;y)$ and $\Omega_l(W_l)$ are a risk function and regularization terms, respectively. L is the number of layers.

Problem Relaxation

The MLP training problem contains multiple constraints, and hence is relaxed via imposing ℓ_2 penalties on the objective as follows:

Problem (Relaxed MLP training problem)

$$\min_{W_l,b_l,z_l,a_l} F(\mathrm{W},\mathrm{b},\mathrm{z},\mathrm{a}) = R(z_L;y) + \sum\nolimits_{l=1}^{L} \Omega_l(W_l)$$

$$+ (\nu/2) \sum\nolimits_{l=1}^{L-1} (\|z_l - W_l a_{l-1} - b_l\|_2^2 + \|a_l - f_l(z_l)\|_2^2)$$

$$s.t.\ z_L = W_L a_{L-1} + b_L$$

where $\mathbf{W} = \{W_l\}_{l=1}^{L}$, $\mathbf{b} = \{b_l\}_{l=1}^{L}$, $\mathbf{z} = \{z_l\}_{l=1}^{L}$, $\mathbf{a} = \{a_l\}_{l=1}^{L-1}$, $\nu > 0$ is a tuning parameter. As $\nu \to \infty$, the relaxed problem approaches the original problem. Then the augmented Lagrangian is formulated and subproblems are solved alternately.

Challenges of the Existing ADMM Work

- **Slow convergence towards solutions**.
  The ADMM usually converges slowly to high accuracy, even for simple examples. It is often the case that ADMM becomes trapped in a modest solution.
- **Cubic time complexity with regard to feature dimensions**.
  The implementation of the ADMM is very time-consuming for real-world datasets. Previous experiments showed that ADMM required more than 7000 cores to train a neural network with just 300 neurons [1]. This computational bottleneck mainly originates from the matrix inversion, whose time complexity is approximately $O(n^3)$.
- **The lack of convergence guarantees**.
  The convergence theory of the nonconvex ADMM cannot be directly applied to deep learning problem. This is because a typical deep learning problem consists of a combination of linear and nonlinear mappings, causing optimization problems to be highly nonconvex.

Table of Contents

1. Background Introduction

2. The dlADMM Algorithm

3. The pdADMM Algorithm

dlADMM: Outline

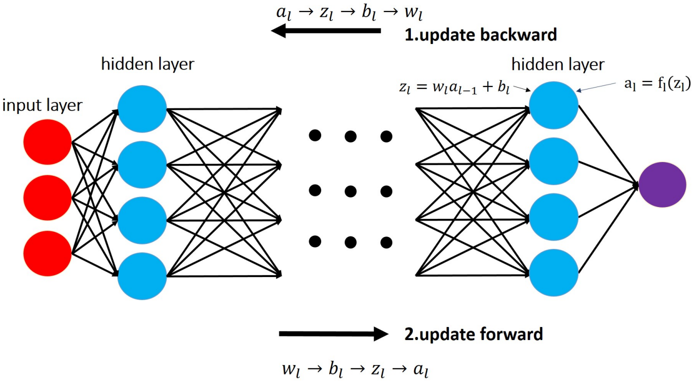
In order to deal with these challenges, we propose a deep learning Alternating Direction Method of Multipliers(dlADMM) [2]. Specifically,

Challenge	Contribution
Slow convergence	New update routine.
$O(n^3)$ time complexity	Reduce to $O(n^2)$ by quadratic approximation.
The lack of convergence guarantees	Proofs of convergence to a critical point.

Table: The improvement of the dlADMM algorithm.

dlADMM: Update Parameters Backward and then Forward

We propose a new update routine: the parameter information for all layers can be exchanged efficiently by updating parameters backward and then forward.



$a_l \rightarrow z_l \rightarrow b_l \rightarrow w_l$
1.update backward
hidden layer
hidden layer
input layer
$z_l = w_l a_{l-1} + b_l$
$a_l = f_l(z_l)$
2.update forward
$w_l \rightarrow b_l \rightarrow z_l \rightarrow a_l$

dlADMM: The Augmented Lagrangian

The Augmented Lagrangian is formulated mathematically as follows:

$$L_\rho(\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{a},u) = R(z_L;y) + \sum\nolimits_{l=1}^{L} \Omega_l(W_l) + \phi(\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{a},u)$$

where $\phi(\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{a},u) = (\nu/2)\sum_{l=1}^{L-1}(\|z_l - W_l a_{l-1} - b_l\|_2^2 + \|a_l - f_l(z_l)\|_2^2) + u^T(z_L - W_L a_{L-1} - b_L) + (\rho/2)\|z_L - W_L a_{L-1} - b_L\|_2^2$, u is a dual variable and $\rho > 0$ is a hyperparameter. We denote $\overline{W}_l^{k+1}$, $\overline{b}_l^{k+1}$, $\overline{z}_l^{k+1}$ and $\overline{a}_l^{k+1}$ as the backward update of the dlADMM for the l-th layer in the $(k+1)$-th iteration , while W_l^{k+1}, b_l^{k+1}, z_l^{k+1} and a_l^{k+1} are denoted as the forward update of the dlADMM for the l-th layer in the $(k+1)$-th iteration.

dlADMM: Pseudocode

Algorithm 1 The dlADMM Algorithm

Require: y, $a_0 = x$, ρ, ν, $k = 0$.

Ensure: $\mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{a}$

```
while $\mathbf{W}^{k+1}, \mathbf{b}^{k+1}, \mathbf{z}^{k+1}, \mathbf{a}^{k+1}$ not converged do
    Update $\overline{z}_L^{k+1}$, $\overline{b}_L^{k+1}$ and $\overline{W}_L^{k+1}$ in order.
    for $l = L-1$ to 1 do
        Update $\overline{a}_l^{k+1}$, $\overline{z}_l^{k+1}$, $\overline{b}_l^{k+1}$, and $\overline{W}_l^{k+1}$ in order.
    end for
    for $l = 1$ to $L-1$ do
        Update $W_l^{k+1}$, $b_l^{k+1}$, $z_l^{k+1}$, $a_l^{k+1}$ in order.
    end for
    Update $W_L^{k+1}$, $b_L^{k+1}$, and $z_L^{k+1}$ in order.
    $r^{k+1} \leftarrow z_L^{k+1} - W_L^{k+1} a_{L-1}^{k+1} - b_L^{k+1}$.#Compute Residual.
    $u^{k+1} \leftarrow u^k + \rho r^{k+1}$.
    $k \leftarrow k+1$.
end while
Output $\mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{a}$.
```

Backward Update

Forward Update

dlADMM: Quadratic Approximation(1)

In order to avoid matrix inversion, we apply the quadratic approximation techniques to subproblems. Take W_l^{k+1} as an example:

$$\begin{aligned} W_l^{k+1} \leftarrow \arg\min\nolimits_{W_l} \phi(\{W_i^{k+1}\}_{i=1}^{l-1}, W_l, \{\overline{W}_i^{k+1}\}_{i=l+1}^{L}, \mathbf{b}_{l-1}^{k+1}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k) \\ + \Omega(W_l) \end{aligned}$$

We define $P_l(W_l; \theta_l^{k+1})$ as a quadratic approximation of ϕ, which is mathematically reformulated as follows:

$$\begin{aligned} P_l(W_l; \theta_l^{k+1}) &= \phi(\mathbf{W}_{l-1}^{k+1}, \mathbf{b}_{l-1}^{k+1}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k) \\ &+ (\nabla_{\overline{W}_l^{k+1}} \phi)^T(\mathbf{W}_{l-1}^{k+1}, \mathbf{b}_{l-1}^{k+1}, \mathbf{z}_{l-1}^{k+1}, \mathbf{a}_{l-1}^{k+1}, u^k)(W_l - \overline{W}_l^{k+1}) \\ &+ \|\theta_l^{k+1} \circ (W_l - \overline{W}_l^{k+1})^{\circ 2}\|_1/2 \end{aligned}$$

where $\theta_l^{k+1} > 0$ is a parameter vector, which can be chosen by backtracking to satisfy the condition
$\phi(\mathbf{W}_l^{k+1}, \mathbf{b}_l^{k+1}, \mathbf{z}_l^{k+1}, \mathbf{a}_l^{k+1}, u^k) \leq P_l(W_l^{k+1}; \theta_l^{k+1})$.

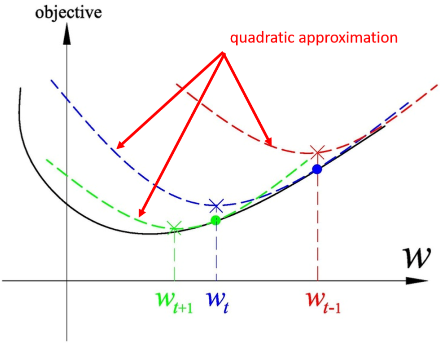
dlADMM: Quadratic Approximation(2)

We instead solve the following problem:

$$W_l^{k+1} \leftarrow \arg\min_{W_l} P_l(W_l;\theta_l^{k+1}) + \Omega_l(W_l)$$

which is convex and may have a closed-form solution for ℓ_1 or ℓ_2 regularization.

The geometric interpretation of quadratic approximation is shown in the below figure:



dlADMM: Convergence Assumptions

Assumption (Closed-form Solution)

There exist activation functions $a_l = f_l(z_l)$ such that $\overline{z}_l^{k+1}$-subproblem and z_l^{k+1}-subproblem have closed form solutions $\overline{z}_l^{k+1} = \overline{h}(\overline{\mathrm{W}}_{l+1}^{k+1}, \overline{\mathrm{b}}_{l+1}^{k+1}, \overline{\mathrm{a}}_l^{k+1})$ and $z_l^{k+1} = h(\mathrm{W}_l^{k+1}, \mathrm{b}_l^{k+1}, \mathrm{a}_{l-1}^{k+1})$, respectively, where $\overline{h}(\bullet)$ and $h(\bullet)$ are continuous functions.

This assumption can be satisfied by common activation functions such as ReLU and leaky ReLU.

Assumption (Objective Function)

$F(\mathrm{W}, \mathrm{b}, \mathrm{z}, \mathrm{a})$ is coercive over the nonempty set $G = \{(\mathrm{W}, \mathrm{b}, \mathrm{z}, \mathrm{a}) : z_L - W_L a_{L-1} - b_L = 0\}$. In other words, $F(\mathrm{W}, \mathrm{b}, \mathrm{z}, \mathrm{a}) \to \infty$ if $(\mathrm{W}, \mathrm{b}, \mathrm{z}, \mathrm{a}) \in G$ and $\|(\mathrm{W}, \mathrm{b}, \mathrm{z}, \ \mathrm{a})\| \to \infty$. Moreover, $R(z_L; y)$ is Lipschitz differentiable with Lipschitz constant $H \geq 0$.

The cross-entropy loss and the least square loss are Lipschitz differentiable.

dlADMM: Convergence Properties(1)

If two assumptions hold, we can prove three convergence properties as follows:

Property (Boundness)

If $\rho > 2H$*, then* $\{\mathrm{W}^k, \mathrm{b}^k, \mathrm{z}^k, \mathrm{a}^k, u^k\}$ *is bounded, and* $L_\rho(\mathrm{W}^k, \mathrm{b}^k, \mathrm{z}^k, \mathrm{a}^k, u^k)$ *is lower bounded.*

Property (Sufficient Descent)

If $\rho > 2H$ *so that* $C_1 = \rho/2 - H/2 - H^2/\rho > 0$*, then there exists* C_2 *such that*

$$\begin{aligned}
&L_\rho(\mathrm{W}^k, \mathrm{b}^k, \mathrm{z}^k, \mathrm{a}^k, u^k) - L_\rho(\mathrm{W}^{k+1}, \mathrm{b}^{k+1}, \mathrm{z}^{k+1}, \mathrm{a}^{k+1}, u^{k+1})\\
&\geq C_2(\textstyle\sum_{l=1}^{L}(\|\overline{W}_l^{k+1} - W_l^k\|_2^2 + \|W_l^{k+1} - \overline{W}_l^{k+1}\|_2^2\\
&+ \|\overline{b}_l^{k+1} - b_l^k\|_2^2 + \|b_l^{k+1} - \overline{b}_l^{k+1}\|_2^2) + \textstyle\sum_{l=1}^{L-1}(\|\overline{a}_l^{k+1} - a_l^k\|_2^2\\
&+ \|a_l^{k+1} - \overline{a}_l^{k+1}\|_2^2) + \|\overline{z}_L^{k+1} - z_L^k\|_2^2 + \|z_L^{k+1} - \overline{z}_L^{k+1}\|_2^2)
\end{aligned}$$

dlADMM: Convergence Properties(2)

The first two properties guarantee the convergence of the augmented Lagrangian, and the third one guarantees the boundness of subgradient:

Property (Subgradient Bound)

There exist a constant $C > 0$ and $g \in \partial L_\rho(\mathrm{W}^{k+1}, \mathrm{b}^{k+1}, \mathrm{z}^{k+1}, \mathrm{a}^{k+1})$ such that

$$\begin{aligned} \|g\| \leq C(\|\mathrm{W}^{k+1} - \overline{\mathrm{W}}^{k+1}\| + \|\mathrm{b}^{k+1} - \overline{\mathrm{b}}^{k+1}\| \\ + \|\mathrm{z}^{k+1} - \overline{\mathrm{z}}^{k+1}\| + \|\mathrm{a}^{k+1} - \overline{\mathrm{a}}^{k+1}\| + \|\mathrm{z}^{k+1} - \mathrm{z}^{k}\|) \end{aligned}$$

Based on three convergence properties, we can prove the convergence of the dlADMM algorithm to a critical point with a sublinear convergence rate $o(1/k)$.

dlADMM: Convergence to a Critical Point Sublinearly

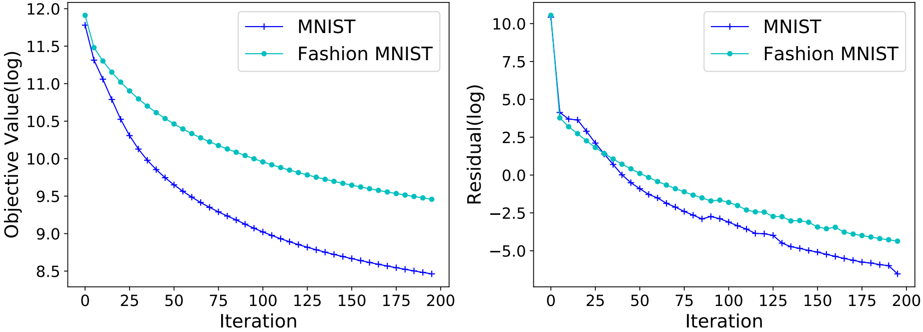
Theorem (Convergence to a Critical Point)

If $\rho > 2H$, then for the variables $(\mathrm{W}, \mathrm{b}, \mathrm{z}, \mathrm{a}, u)$ in the relaxed MLP training problem, starting from any $(\mathrm{W}^0, \mathrm{b}^0, \mathrm{z}^0, \mathrm{a}^0, u^0)$, it has at least a limit point $(\mathrm{W}^*, \mathrm{b}^*, \mathrm{z}^*, \mathrm{a}^*, u^*)$, and any limit point $(\mathrm{W}^*, \mathrm{b}^*, \mathrm{z}^*, \mathrm{a}^*, u^*)$ is a critical point. That is, $0 \in \partial L_\rho(\mathrm{W}^*, \mathrm{b}^*, \mathrm{z}^*, \mathrm{a}^*, u^*)$.

Theorem (Convergence Rate)

For a sequence $(\mathrm{W}^k, \mathrm{b}^k, \mathrm{z}^k, \mathrm{a}^k, u^k)$, define $c_k = \min_{0 \leq i \leq k}(\sum_{l=1}^{L}(\|\overline{W}_l^{i+1} - W_l^i\|_2^2 + \|W_l^{i+1} - \overline{W}_l^{i+1}\|_2^2 + \|\overline{b}_l^{i+1} - b_l^i\|_2^2 + \|b_l^{i+1} - \overline{b}_l^{i+1}\|_2^2) + \sum_{l=1}^{L-1}(\|\overline{a}_l^{i+1} - a_l^i\|_2^2 + \|a_l^{i+1} - \overline{a}_l^{i+1}\|_2^2) + \|\overline{z}_L^{i+1} - z_L^i\|_2^2 + \|z_L^{i+1} - \overline{z}_L^{i+1}\|_2^2)$, then the convergence rate of c_k is $o(1/k)$.

Experimental Results: Convergence

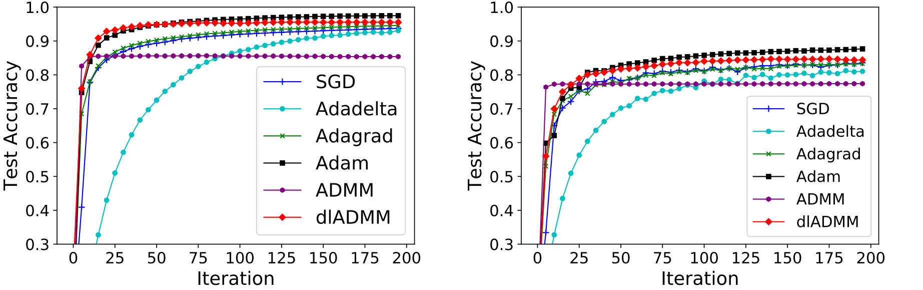


(a). Objective value

(b). Residual

Figure: Convergence curves of dlADMM algorithm for MNIST and Fashion MNIST datasets when $\rho = 1$: the dlADMM algorithm converged.

Experimental Results: Performance



(a). MNIST dataset.

(b). Fashion-MNIST dataset.

Figure: Test Performance of all methods for two datasets: the dlADMM algorithm outperformed most of the comparison methods.

Table of Contents

1. Background Introduction
2. The dlADMM Algorithm
3. The pdADMM Algorithm

Backward Locking

The backward locking problem restrict the parallel training of SGD. For example, Module A can not finish Step 6 until Module B finishes Step 5.

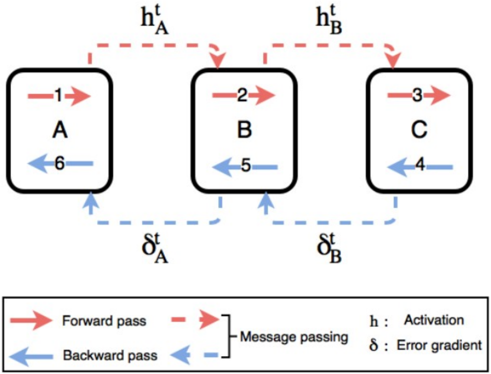
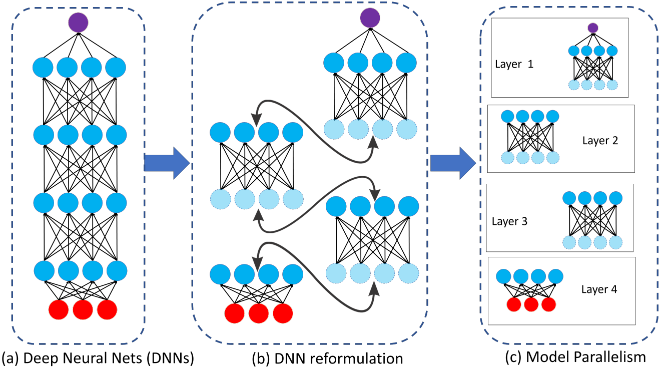


Figure: The mechanism of backpropagation.[1]

[1] This figure is borrowed from Huo et al. ”Decoupled parallel backpropagation with convergence guarantee.” ICML 2018.

pdADMM: Break the Backward Locking by Layer Splitting

The parallel deep learning Alternating Direction Method of Multipliers (pdADMM) can address the backward locking problem[3]. Specifically, by splitting the whole neural network into multiple layers, each of which can be optimized by an independent worker. Therefore, the layerwise training can be implemented in parallel.



pdADMM: Problem Formulation

Problem (MLP Training Problem Revisited)

$$\min_{W_l,b_l,z_l,a_l} R(z_L;y)$$
$$s.t. z_l = W_l a_{l-1} + b_l (l = 1,\cdots,L),\ \ a_l = f_l(z_l)\ (l = 1,\cdots,L-1)$$

Notice that a_l is both the input for the l-th layer and the output for the $(l-1)$-th layer. Let p_l and q_l be the input and output for the l-th layer, respectively. Then the MLP training problem can be relaxed to

Problem (Parallel MLP Training Problem)

$$\min_{\mathrm{p},\mathrm{W},\mathrm{b},\mathrm{z},\mathrm{q}} F(\mathrm{p},\mathrm{W},\mathrm{b},\mathrm{z},\mathrm{q}) = R(z_L;y)$$
$$+ (\nu/2)(\sum\nolimits_{l=1}^{L} \|z_l - W_l p_l - b_l\|_2^2 + \sum\nolimits_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2)$$
$$s.t.\ p_{l+1} = q_l$$

where $\mathbf{p} = \{p_l\}_{l=1}^{L}$, $\mathbf{W} = \{W_l\}_{l=1}^{L}$, $\mathbf{b} = \{b_l\}_{l=1}^{L}$, $\mathbf{z} = \{z_l\}_{l=1}^{L}$, $\mathbf{q} = \{q_l\}_{l=1}^{L-1}$, and $\nu > 0$ is a tuning parameter.

The Augmented Lagrangian

The Augmented Lagrangian is formulated mathematically as follows:

$$
\begin{aligned}
&L_\rho(\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q},\mathbf{u})\\
&= F(\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q}) + \sum\nolimits_{l=1}^{L-1} (u_l^T(p_{l+1}-q_l) + (\rho/2)\|p_{l+1}-q_l\|_2^2)\\
&= R(z_L;y) + \phi(p_1,W_1,b_1,z_1) + \sum\nolimits_{l=2}^{L} \phi(p_l,W_l,b_l,z_l,q_{l-1},u_{l-1})\\
&+ (\nu/2)\sum\nolimits_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2
\end{aligned}
$$

where $\phi(p_1,W_1,b_1,z_1) = (\nu/2)\|z_1 - W_1 p_1 - b_1\|_2^2$, $\phi(p_l,W_l,b_l,z_l,q_{l-1},u_{l-1}) =$ $(\nu/2)\|z_l - W_l p_l - b_l\|_2^2 + u_{l-1}^T(p_l - q_{l-1}) + (\rho/2)\|p_l - q_{l-1}\|_2^2$, $u_l(l=1,\cdots,L-1)$ are dual variables, $\rho > 0$ is a hyperparameter, and $\mathbf{u} = \{u_l\}_{l=1}^{L-1}$. Subproblems are solved by the quadratic approximation techniques and backtracking.

pdADMM: Pseudocode

Algorithm 2 the pdADMM Algorithm

```
Require: $y$, $p_1 = x$, $\rho$, $\nu$.
Ensure: $\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}$.
Initialize $k = 0$.
while $\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k$ not converged do
    Update $p_l^{k+1}$ of different $l$ in parallel.
    Update $W_l^{k+1}$ of different $l$ in parallel.
    Update $b_l^{k+1}$ of different $l$ in parallel.
    Update $z_l^{k+1}$ of different $l$ in parallel.
    Update $q_l^{k+1}$ of different $l$ in parallel.
    $r_l^k \leftarrow p_{l+1}^{k+1} - q_l^{k+1} (l = 1, \cdots, L)$ in parallel # Compute residuals.
    $u_l^{k+1} \leftarrow u_l^k + \rho r_l^k$ of different $l$ in parallel.
    $k \leftarrow k + 1$.
end while
Output $\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}$.
```

pdADMM: Convergence Assumptions

Assumption

$f_l(z_l)$ is Lipschitz continuous with coefficient $S > 0$, and $F(\mathrm{p}, \mathrm{W}, \mathrm{b}, \mathrm{z}, \mathrm{q})$ is coercive. Moreover, $\partial f_l(z_l)$ is bounded, i.e. there exists $M > 0$ such that $\|\partial f_l(z_l)\| \leq M$.

The convergence conditions of the pdADMM algorithm are milder than those of the dlADMM algorithm.

Algorithm	dlADMM	pdADMM
Objective	Coercive	Coercive
Activation $f_l(z_l)$	ReLU and leaky ReLU	ReLU, leaky ReLU, sigmoid and tanh
Loss $R(z_L; y)$	Lipschitz differentiable	No assumption

Table: Convergence Assumptions between dlADMM and pdADMM

pdADMM: Convergence Results

We can prove the convergence of the pdADMM to a critical point using the similar proofs as dlADMM, which is shown as follows:

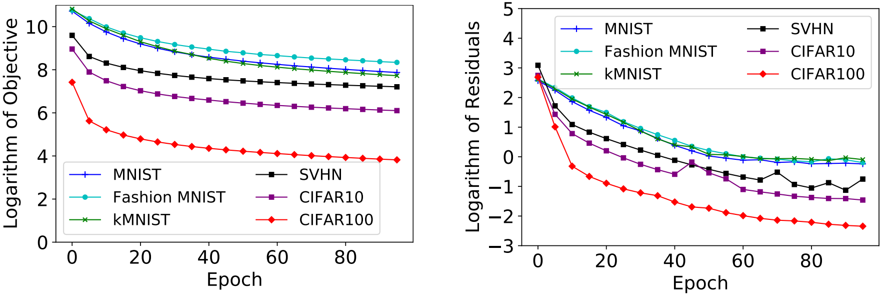
Theorem (Convergence to a Critical Point)

If $\rho > \max(4\nu S^2, (\sqrt{17}+1)\nu/2)$*, then for the variables* $(\mathrm{p}, \mathrm{W}, \mathrm{b}, \mathrm{z}, \mathrm{q}, \mathrm{u})$ *in the parallel MLP training problem, starting from any* $(\mathrm{p}^0, \mathrm{W}^0, \mathrm{b}^0, \mathrm{z}^0, \mathrm{q}^0, \mathrm{u}^0)$*,* $(\mathrm{p}^k, \mathrm{W}^k, \mathrm{b}^k, \mathrm{z}^k, \mathrm{q}^k, \mathrm{u}^k)$ *has at least a limit point* $(\mathrm{p}^*, \mathrm{W}^*, \mathrm{b}^*, \mathrm{z}^*, \mathrm{q}^*, \mathrm{u}^*)$*, and any limit point is a critical point of parallel MLP training problem. That is,* $0 \in \partial L_\rho(\mathrm{p}^*, \mathrm{W}^*, \mathrm{b}^*, \mathrm{z}^*, \mathrm{q}^*, \mathrm{u}^*)$*.*

Theorem (Convergence Rate)

For a sequence $(\mathrm{p}^k, \mathrm{W}^k, \mathrm{b}^k, \mathrm{z}^k, \mathrm{q}^k, \mathrm{u}^k)$*, define* $c_k = \min_{0\leq i\leq k}(\sum_{l=2}^{L}(\tau_l^{i+1}/2)\|p_l^{i+1} - p_l^i\|_2^2 + \sum_{l=1}^{L}(\theta_l^{i+1}/2)\|W_l^{i+1} - W_l^i\|_2^2 + \sum_{l=1}^{L}(\nu/2)\|b_l^{i+1} - b_l^i\|_2^2 + \sum_{l=1}^{L-1} C_1\|z_l^{i+1} - z_l^i\|_2^2 + (\nu/2)\|z_L^{i+1} - z_L^i\|_2^2 + \sum_{l=1}^{L-1} C_2\|q_l^{i+1} - q_l^i\|_2^2)$ *where* $C_1 = \nu/2 - 2\nu^2 S^2/\rho > 0$ *and* $C_2 = \rho/2 - 2\nu^2/\rho - \nu/2 > 0$*, then the convergence rate of* c_k *is* $o(1/k)$*.*

Experimental Results: Convergence



(a). Objective versus epoch

(b). Residual versus epoch

Figure: The convergence of the proposed pdADMM: the objective decreases monotonously, and the residual converges to 0.

Experimental Results: Speedup

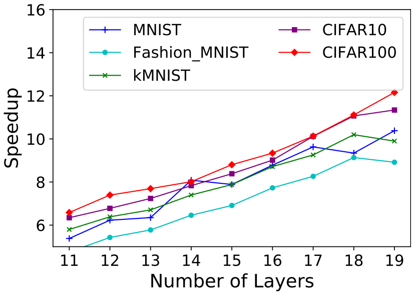
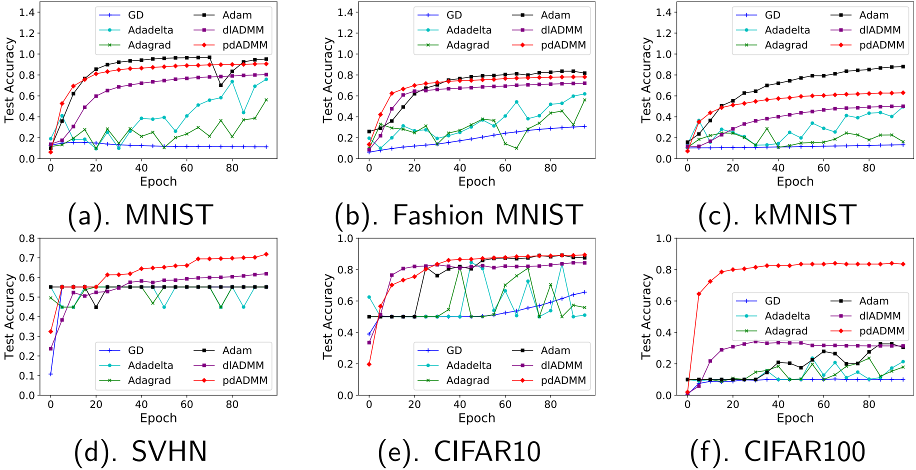


Figure: The relationship between speedup and the number of layers: the speedup increases linearly with the number of layers.

Neurons#	Serial pdADMM (sec)	pdADMM(sec)	Speedup
1500	237.66	26.78	8.87
1600	348.70	31.78	10.97
1700	390.51	35.79	10.91
1800	475.60	41.37	11.50
1900	465.57	45.87	10.15
2000	570.90	50.70	11.26
2000	570.9	50.7	11.26
2100	570	54.91	10.38
2200	678.83	63.59	10.68
2300	710.3	70.36	10.10
2400	766.82	62.5	12.27

Table: The relation between speedup and number of neurons on the MNIST dataset: the pdADMM runs 10 times faster than its serial version.

Experimental Results: Performance



(a). MNIST

(b). Fashion MNIST

(c). kMNIST

(d). SVHN

(e). CIFAR10

(f). CIFAR100

Figure: Test accuracy of all methods: pdADMM outperformed most comparison methods.

Code Release

dlADMM: `https://github.com/xianggebenben/dlADMM`
pdADMM: `https://github.com/xianggebenben/pdADMM`

Thank you for your attention!

Reference

- G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, “Training neural networks without gradients: A scalable admm approach,” in *International conference on machine learning*. PMLR, 2016, pp. 2722–2731.
- J. Wang, F. Yu, X. Chen, and L. Zhao, “Admm for efficient deep learning with global convergence,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 111–119.
- J. Wang, Z. Chai, Y. Cheng, and L. Zhao, “Toward model parallelism for deep neural network based on gradient-free admm framework,” in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 591–600.