Accelerated Gradient-free Neural Network Training by Multi-convex Alternating Optimization

Junxiang Wang
Joint works with Hongyi Li (Xidian), and Liang Zhao (Emory)

Department of Computer Science and Informatics, Emory University

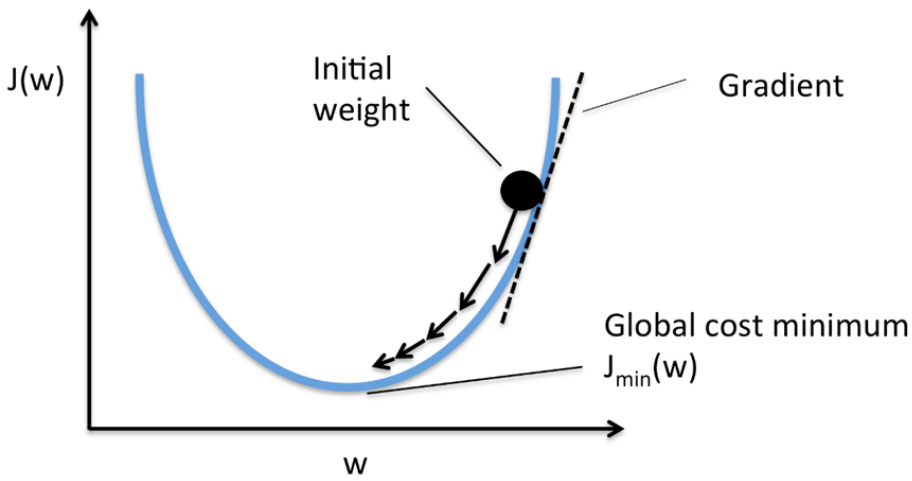
EURO 2022

Table of Contents

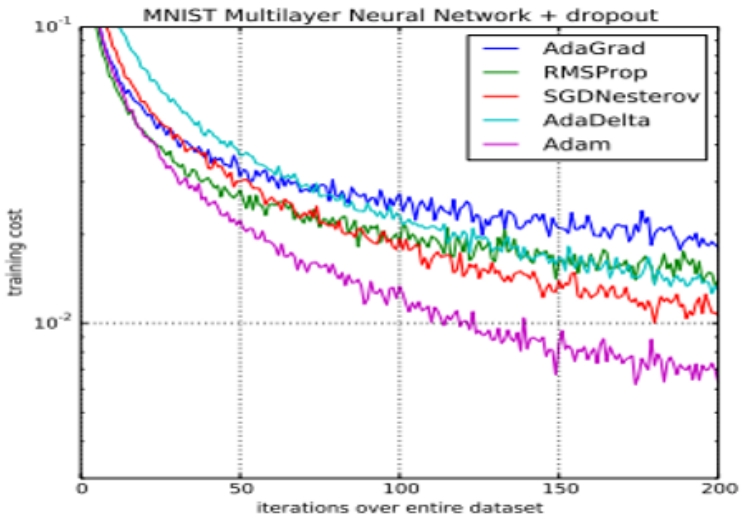
Table of Contents

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)

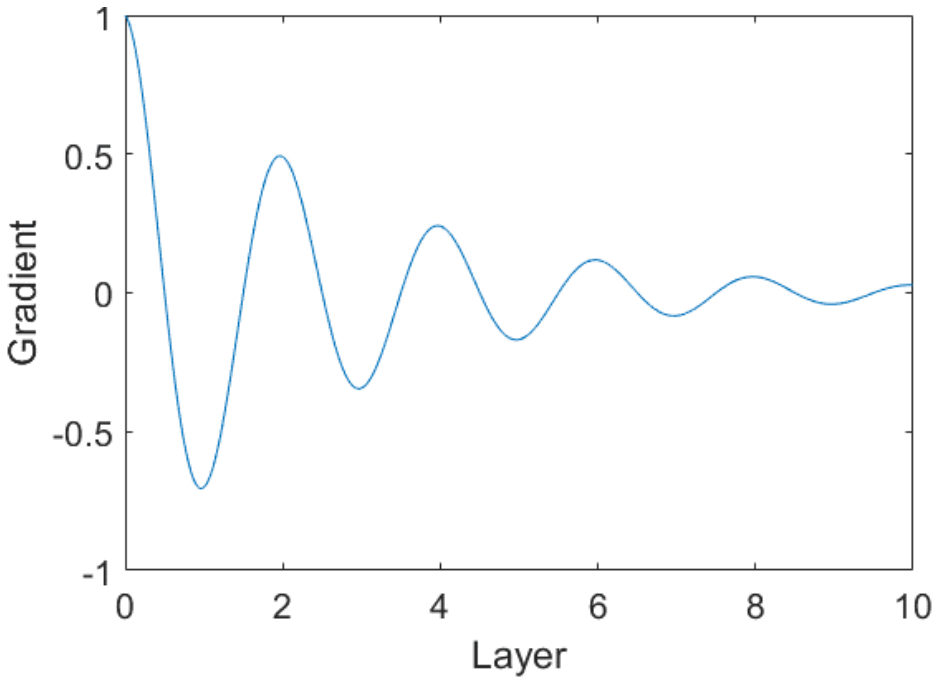


Outstanding Performance

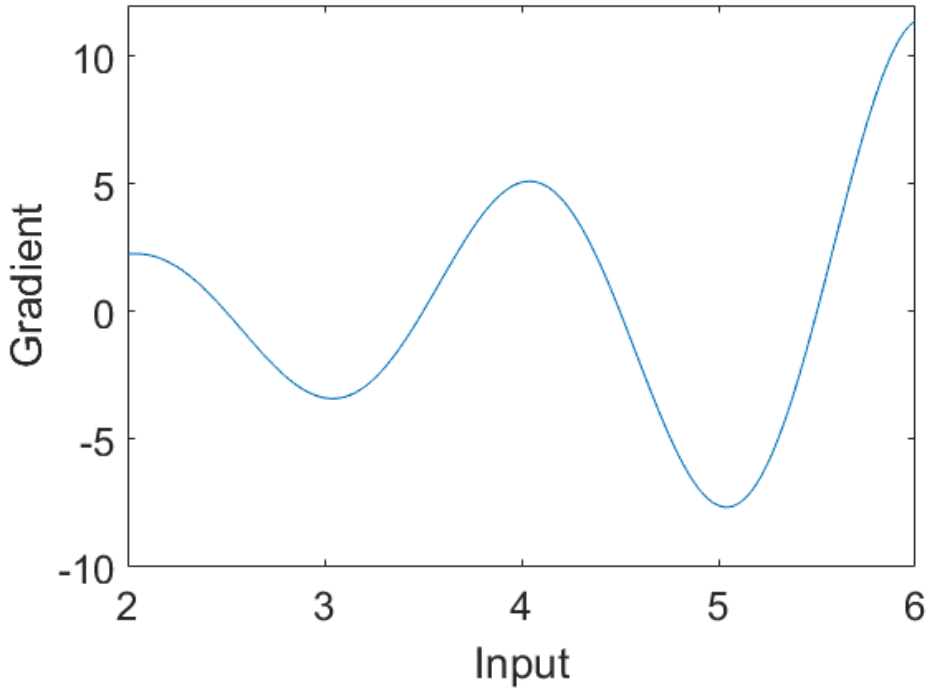
Challenges of SGD

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 Alternating Minimization (AM)

The AM has following advantages over SGD:

- **The potentials to decompose complex objectives into simple subproblems.**
  A neural network problem is reformulated as a nested function associated with multiple linear and nonlinear transformations across multi-layers. This nested structure is then decomposed into a series of linear and nonlinear equality constraints by introducing auxiliary variables and penalty hyperparameters, which generate multiple subproblems and can be minimized alternately.
- **Be immune to gradient vanishing.**
  The AM splits a neural network into layerwise components, and prevents the accumulative calculation of gradients.

Existing Works on Training Neural Network using AM(1)

As a good representative of AM, the Alternating Direction Method of Multipliers (ADMM) has received attention from the deep learning community. Taylor et al. [1] deals with the following Multi-Layer Perceptron (MLP) model:

Problem (MLP Training Problem)

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

where $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ is a weight matrix for the l-th layer. 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 h_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.

Existing Works on Training Neural Network using AM(2)

Our recent work [2] improved the performance and the efficiency of the ADMM via the backward-forward training and the quadratic approximation with backtracking. Moreover, we proved that the ADMM converges to a stationary point of the problem.
However, two challenges exist for existing AM approaches:
1. Convergence properties are sensitive to penalty parameters. For example, the convergence guarantees mentioned above do not hold when the hyperparameters are small.
2. Slow convergence rate. Almost all existing AM methods can only achieve a sublinear convergence rate. For example, the convergence rate of the ADMM and the Block Coordinate Descent (BCD) is proven to be $O(1/k)$, where k is the number of iterations.

Table of Contents

mDLAM: Outline

To deal with these two challenges, we propose a monotonous Deep Learning Alternating Minimization (mDLAM). Specifically,

- We propose a novel formulation for neural network optimization. The deeply nested neural network is disentangled into separate functions by inherently convex inequality constraints.
- We present an efficient mDLAM algorithm. The Nesterov acceleration technique is applied to further boost convergence.
- We investigate the convergence of the proposed mDLAM algorithm under mild conditions. It converges to a stationary point whatever hyperparameters we choose with a linear convergence rate.

mDLAM: Inequality Approximation for Deep Learning(1)

In the MLP training problem, the constraints $a_l = h_l(z_l)$ are the most difficult to handle due to two reasons:
1. Nonlinear activation functions make it difficult to obtain optimal solutions when solving subproblems.
2. There is no convergence guarantee for AM methods to solve the MLP training problem with nonlinear constraints.
To deal with these two challenges, we require an assumption for problem transformation. To achieve this, firstly, we define the quasilinearity as follows:

Definition (Quasilinearity)

A function $f(x)$ is quasiconvex if for any sublevel set $S_\nu(f) = \{x|f(x) \leq \nu\}$ is a convex set. Likewise, A function $f(x)$ is quasiconcave if for any superlevel set $S_\nu(f) = \{x|f(x) \geq \nu\}$ is a convex set. A function $f(x)$ is quasilinear if it is both quasiconvex and quasiconcave.

mDLAM: Inequality Approximation for Deep Learning(2)

Based on the definition of quasilinearity, we have the following assumption:

Assumption

$h_l(z_l)$ $(l = 1, \ldots, n)$ are quasilinear.

This assumption is so mild that most common activation functions satisfy it, including tanh, sigmoid, and the Rectified Linear Unit (ReLU).
Then we innovatively transform nonconvex constraints $a_l = h_l(z_l)$ into inequality constraints. To achieve this, we introduce a tolerance $\varepsilon > 0$:

$$\min_{W_l, z_l, a_l} R(z_L; y) + \sum_{l=1}^{L} \Omega_l(W_l),$$
$$s.t.\ z_l = W_l a_{l-1} (l = 1, \cdots, L), h_l(z_l) - \varepsilon \leq a_l \leq h_l(z_l) + \varepsilon (l = 1, \cdots, L-1).$$

The introduction of ε is to project $a_l = h_l(z_l)$ to ε-balls.

mDLAM: Inequality Approximation for Deep Learning(3)

For $z_l = W_l a_{l-1}$, this can be transformed into a penalty term in the objective function to minimize the difference between z_l and $W_l a_{l-1}$.

Problem (Inequality-Approximate MLP Training)

$$\min_{W_l, z_l, a_l} F(\mathbf{W}, \mathbf{z}, \mathbf{a}) = R(z_L; y) + \sum_{l=1}^{L} \Omega_l(W_l) + \sum_{l=1}^{L} \phi(a_{l-1}, W_l, z_l),$$

$$s.t.\ h_l(z_l) - \varepsilon \leq a_l \leq h_l(z_l) + \varepsilon (l = 1, \cdots, L-1).$$

The penalty term is defined as $\phi(a_{l-1}, W_l, z_l) = \frac{\rho}{2}\|z_l - W_l a_{l-1}\|_2^2$, where $\rho > 0$ a penalty parameter. $\mathbf{W} = \{W_l\}_{l=1}^{L}$, $\mathbf{z} = \{z_l\}_{l=1}^{L}$,$\mathbf{a} = \{a_l\}_{l=1}^{L-1}$. As $\rho \to \infty$ and $\varepsilon \to 0$, the approximate problem approaches the original problem.

mDLAM: Nesterov Acceleration

Due to the multi-convexity of F, AM is a natural fit for the approximate problem. In order to accelerate the convergence rate, we utilize the Nesterov acceleration into our algorithm.

Let $s^0 \leftarrow 0$, $s^{k+1} \leftarrow \frac{1+\sqrt{1+4(s^k)^2}}{2}$, $\omega^k \leftarrow \frac{s^k-1}{s^{k+1}}$. Take W_l as an example:

1. $\overline{W}_l^{k+1} \leftarrow W_l^k + (W_l^k - W_l^{k-1})\omega^k$ is an acceleration step, and we minimize the following problem

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

where

$$P_l^{k+1}(W_l; \theta_l^{k+1}) = \phi(a_{l-1}^{k+1}, \overline{W}_l^{k+1}, z_l^k) + (\nabla_{\overline{W}_l^{k+1}} \phi)^T (W_l - \overline{W}_l^{k+1}) + \frac{\theta_l^{k+1}}{2} \| W_l - \overline{W}_l^{k+1} \|_2^2$$

is a quadratic approximation of ϕ, and $\theta_l^{k+1} > 0$ is a scalar parameter.

2. We check whether W_l^{k+1} decreases F. If not, this suggests that the algorithm needs to steps back. To achieve this, we let $\overline{W}_l^{k+1} \leftarrow W_l^k$ and solve Equation (1) again.

The same procedure is applied to other subproblems.

mDLAM: Pseudocode(1)

Algorithm The mDLAM algorithm

Require: y, $a_0 = x$.
Ensure: $a_l, W_l, z_l (l = 1, \cdots, L)$.

Initialize ρ, $k = 0$. $s^0 = 0$.
repeat
  $s^{k+1} \leftarrow \frac{1+\sqrt{1+4(s^k)^2}}{2}$
  $\omega^k \leftarrow \frac{s^k - 1}{s^{k+1}}$
  for $l = 1$ to L **do**
    $\overline{W}_l^{k+1} \leftarrow W_l^k + (W_l^k - W_l^{k-1})\omega^k$ and update W_l^{k+1}.
    if $F(\mathbf{W}_{\leq l}^{k+1}, \mathbf{z}_{\leq l-1}^{k+1}, \mathbf{a}_{\leq l-1}^{k+1}) \geq F(\mathbf{W}_{\leq l-1}^{k+1}, \mathbf{z}_{\leq l-1}^{k+1}, \mathbf{a}_{\leq l-1}^{k+1})\{\#W_l^{k+1}$ increases the objective $F\}$ **then**
      $\overline{W}_l^{k+1} \leftarrow W_l^k$ and update W_l^{k+1}.
    end if
    $\overline{z}_l^{k+1} \leftarrow z_l^k + (z_l^k - z_l^{k-1})\omega^k$
    if $l = L$ **then**
      Update z_L^{k+1}.
      if $F(\mathbf{W}_{\leq L}^{k+1}, \mathbf{z}_{\leq L}^{k+1}, \mathbf{a}_{\leq L-1}^{k+1}) \geq F(\mathbf{W}_{\leq L}^{k+1}, \mathbf{z}_{\leq L-1}^{k+1}, \mathbf{a}_{\leq L-1}^{k+1})\{\#z_L^{k+1}$ increases the objective $F\}$ **then**
        $\overline{z}_L^{k+1} \leftarrow z_L^k$ and update z_L^{k+1}.
      end if

mDLAM: Pseudocode(2)

16: **else**
17: Update z_l^{k+1}.
18: **if** $F(\mathbf{W}_{\leq l}^{k+1}, \mathbf{z}_{\leq l}^{k+1}, \mathbf{a}_{\leq l-1}^{k+1}) \geq F(\mathbf{W}_{\leq l}^{k+1}, \mathbf{z}_{\leq l-1}^{k+1}, \mathbf{a}_{\leq l-1}^{k+1})\{\#z_l^{k+1}$ increases the objective $F\}$ **then**
19: $\overline{z}_l^{k+1} \leftarrow z_l^k$ and update z_l^{k+1}.
20: **end if**
21: $\overline{a}_l^{k+1} \leftarrow a_l^k + (a_l^k - a_l^{k-1})\omega^k$ and update a_l^{k+1}.
22: **if** $F(\mathbf{W}_{\leq l}^{k+1}, \mathbf{z}_{\leq l}^{k+1}, \mathbf{a}_{\leq l}^{k+1}) \geq F(\mathbf{W}_{\leq l}^{k+1}, \mathbf{z}_{\leq l}^{k+1}, \mathbf{a}_{\leq l-1}^{k+1})\{\#a_l^{k+1}$ increases the objective $F\}$ **then**
23: $\overline{a}_l^{k+1} \leftarrow a_l^k$ and update a_l^{k+1}.
24: **end if**
25: **end if**
26: **end for**
27: $k \leftarrow k + 1$.
28: **until** convergence.
29: Output a_l, W_l, z_l.

Lines 6-9, Lines 10-20, and Lines 21-25 solve the W_l-subproblem, the z_l-subproblem, and the a_l-subproblem, respectively.

Table of Contents

Convergence Assumption

Now we provide the theoretical guarantees of the proposed mDLAM algorithm. Before that, coercivity is defined as follows:

Definition (Coercivity)

Any arbitrary function $G(x)$ is coercive over a nonempty set $dom(G)$ if as $\|x\| \to \infty$ and $x \in dom(G)$, we have $G(x) \to \infty$, where $dom(G)$ is a domain set of G.

Then the convergence assumption is shown as follows:

Assumption

$F(\mathbf{W}, \mathbf{z}, \mathbf{a})$ *is coercive over the domain* $\{(\mathbf{W}, \mathbf{z}, \mathbf{a}) | h_l(z_l) - \varepsilon \leq a_l \leq h_l(z_l) + \varepsilon \ (l = 1, \cdots, L-1)\}$.

This assumption is also mild such that common loss functions such as the least square loss and the cross-entropy loss satisfy it.

Convergence Properties(1)

Given the above assumption, three convergence properties hold. The first one is shown as follows:

Lemma (Objective Convergence)

In the mDLAM algorithm, it holds that for any $k \in \mathbb{N}$, $F(\mathbf{W}^k, \mathbf{z}^k, \mathbf{a}^k) \geq F(\mathbf{W}^{k+1}, \mathbf{z}^{k+1}, \mathbf{a}^{k+1})$. Moreover, F is convergent. That is, $F(\mathbf{W}^k, \mathbf{z}^k, \mathbf{a}^k) \to F^*$ as $k \to \infty$, where F^* is the convergent value of F.

This lemma can be proved via the fact that F keeps decreasing after each variable is updated. The convergence of F can be achieved via the monotonous decrease of F directly.

Convergence Properties(2)

Due to the convergence of F, the boundness of the objective and all variables can be ensured via the second convergence property, which is shown as follows:

Lemma (Bounded Objective and Variables)

In the mDLAM algorithm, it holds that for any $k \in \mathbb{N}$
(a). $\mathbf{F}(\mathbf{W}^k, \mathbf{z}^k, \mathbf{a}^k)$ *is upper bounded. Moreover,* $\lim_{k\to\infty} \mathbf{W}^{k+1} - \mathbf{W}^k = 0$, $\lim_{k\to\infty} \mathbf{z}^{k+1} - \mathbf{z}^k = 0$, *and* $\lim_{k\to\infty} \mathbf{a}^{k+1} - \mathbf{a}^k = 0$.
(b). $(\mathbf{W}^k, \mathbf{z}^k, \mathbf{a}^k)$ *is bounded. That is, there exist scalars* $M_{\mathbf{W}}$, $M_{\mathbf{z}}$ *and* $M_{\mathbf{a}}$ *such that* $\|\mathbf{W}^k\| \leq M_{\mathbf{W}}$, $\|\mathbf{z}^k\| \leq M_{\mathbf{z}}$ *and* $\|\mathbf{a}^k\| \leq M_{\mathbf{a}}$.

The boundness of the objective can be proven via the previous lemma, and the boundness of all variables can be obtained via the definition of coercivity and the convergence assumption.

Convergence Properties(3)

The third convergence property ensures that the subgradient of the objective is bounded by all variables, which is shown as follows:

Lemma (Subgradient Bound)

In the mDLAM algorithm, there exist $C_2 = \max(\rho M_{\mathbf{a}}, \rho M_{\mathbf{a}}^2 + \theta_1^{k+1}, \rho M_{\mathbf{a}}^2 + \theta_2^{k+1}, \cdots, \rho M_{\mathbf{a}}^2 + \theta_L^{k+1})$, *and* $g_1^{k+1} \in \partial_{\mathbf{W}^{k+1}} F$ *such that for any* $k \in \mathbb{N}$

$$\|g_1^{k+1}\| \leq C_2(\|\mathbf{W}^{k+1} - \mathbf{W}^k\| + \|\mathbf{z}^{k+1} - \mathbf{z}^k\| + \|\mathbf{W}^k - \mathbf{W}^{k-1}\|).$$

This lemma can be proven via the optimality conditions of all subproblems.

Convergence to a Stationary Point

Now we prove the main convergence results of the proposed mDLAM algorithm based on three convergence properties.
Firstly, the following theorem guarantees that the proposed mDLAM algorithm converges to a stationary point whatever hyperparameters we choose.

Theorem (Convergence to a Stationary Point)

In the mDLAM algorithm, for any $\mathbf{W}$, *any* $\rho > 0$ *and* $\varepsilon > 0$, *starting from any* $\mathbf{W}^0$, *any limit point* $\mathbf{W}^*$ *is a stationary point. That is,* $0 \in \partial_{\mathbf{W}^*} F$.

This can be proven directly from the second and the third convergence properties: from the third convergence property, the subgradient has an upper bound, and it converges to 0 by the second convergence property.

Linear Convergence(1)

Next, we prove the linear convergence of the proposed mDLAM algorithm. Before that, we need to define the local strong convexity as follows:

Definition (Local Strong Convexity)

A function $f(x)$ is locally strongly convex within a bound set $\mathbb{D}$ with a constant μ if

$$f(y) \geq f(x) + g^T(y-x) + \frac{\mu}{2}\|x-y\|_2^2 \ \forall\ g \in \partial f(x) \text{ and } \ x, y \in \mathbb{D}.$$

Simply speaking, a locally strongly convex function lies above a quadratic function within a bounded set. Notice that a locally strongly convex function is not necessarily convex.

Linear Convergence(2)

Given this definition, the linear convergence is shown as follows:

Theorem

In the mDLAM algorithm, if F is locally strongly convex, then for any ρ, there exist $\varepsilon > 0$, $k_1 \in \mathbb{N}$ and $0 < C_1 < 1$ such that it holds for $k > k_1$ that

$$F(\mathbf{W}^{k+1}, \mathbf{z}^{k+1}, \mathbf{a}^{k+1}) - F^* \leq C_1(F(\mathbf{W}^{k-1}, \mathbf{z}^{k-1}, \mathbf{a}^{k-1}) - F^*).$$

Common loss functions such as the square loss and the cross-entropy loss make F locally strongly convex. Notice that ε should satisfy the condition that the objective still decreases after the Nesterov acceleration (i.e. skip Lines 7-8, Lines 13-14, Lines 18-19, and Lines 22-23 in the mDLAM algorithm). The linear convergence then can be proven via the Kurdyka-Lojasiewicz (KL) property. Such a convergence rate is better than a sublinear convergence rate achieved by the ADMM.

Table of Contents

Experimental Settings

An important application of the MLP model is node classification on a graph based on augmented node features. Specifically, given an adjacency matrix A and a node feature matrix H of a graph, we let the k-th augmented feature $X^k = HA^k (k = 0, 1, \cdots, 4)$, which encodes information of graph topology via A^k, and then concatenate them into the input $X = [X_0, \cdots, X_4]$. The MLP model is used to predict the node class based on the input X.

We set up an architecture of three layers, each of which has 100 hidden units. The activation function was set to ReLU. The number of epoch was set to 200.

Datasets

Dataset	Node#	Training Sample#	Test Sample#	Class#	Feature#
Cora	2708	140	1000	7	1433
PubMed	19717	60	1000	3	500
Citeseer	3327	120	1000	6	3703
Coauthor CS	18333	300	1000	15	6805

Table: Four benchmark datasets

Convergence

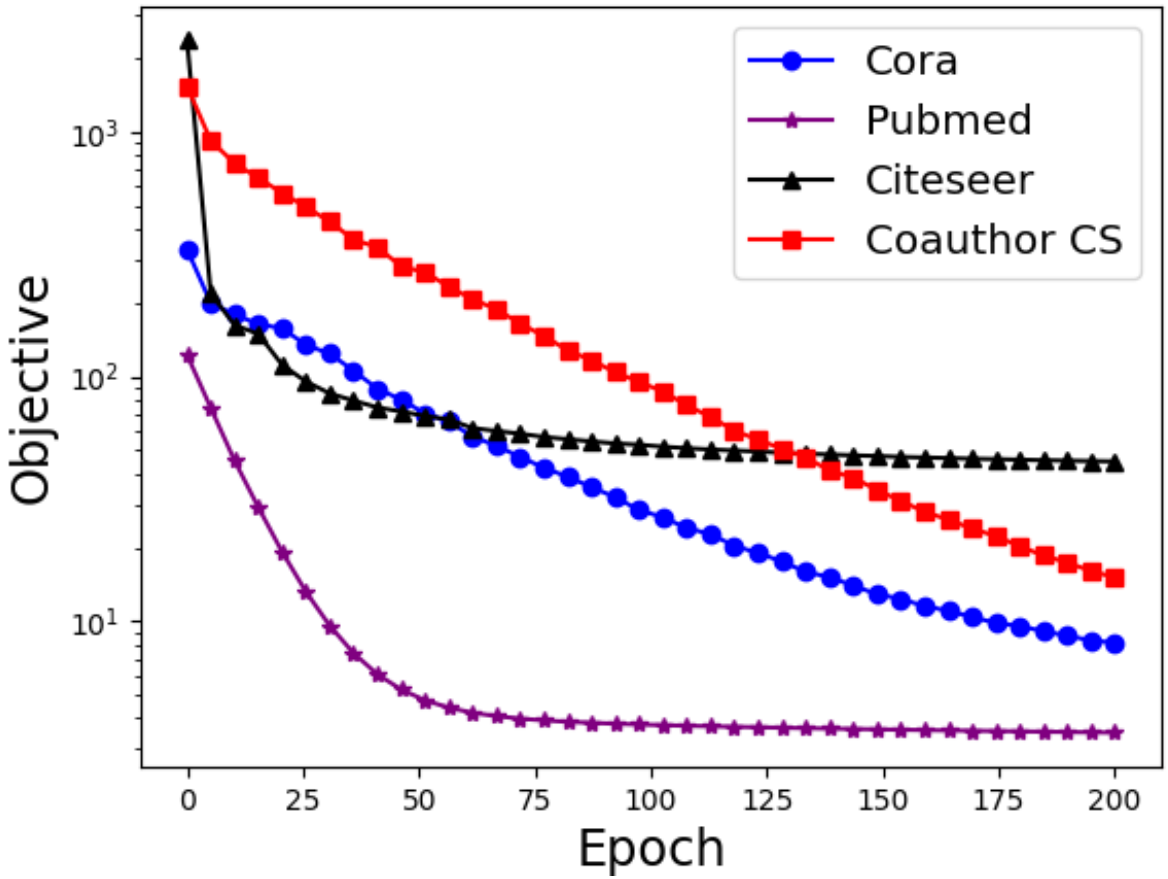
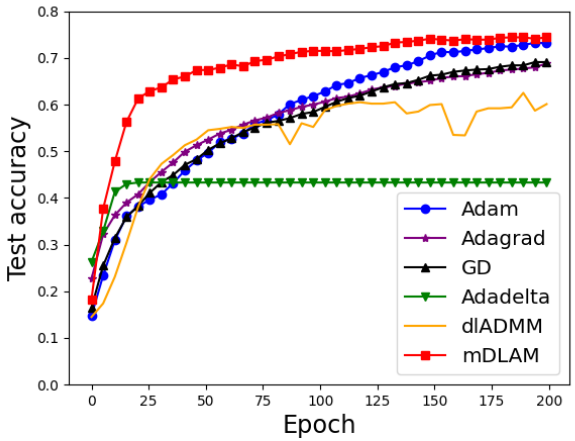
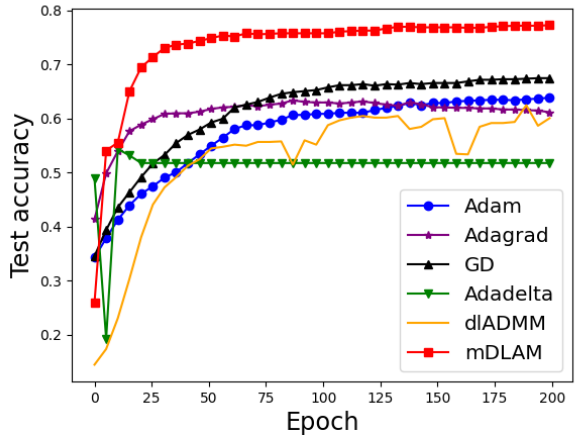


Figure: Convergence curves on four datasets: they all converge linearly when the epoch is larger than 100.

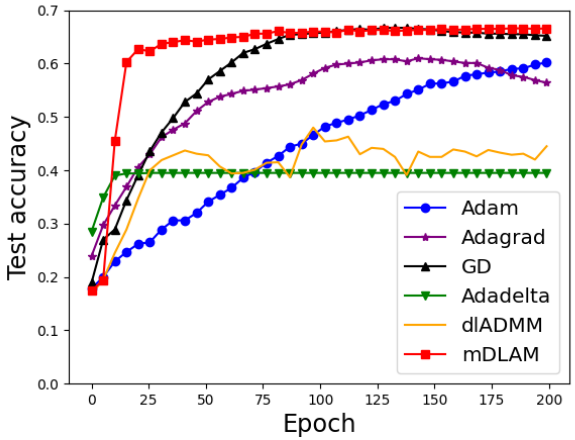
Performance



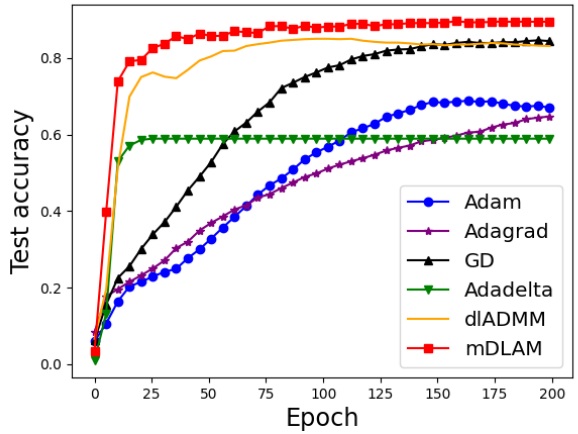
(a). Cora.



(b). Pubmed.



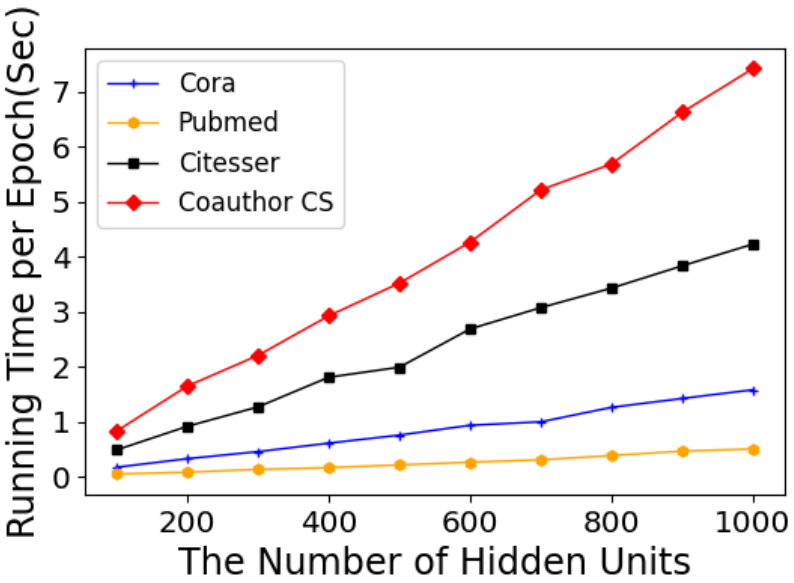
(c). Citeseer.



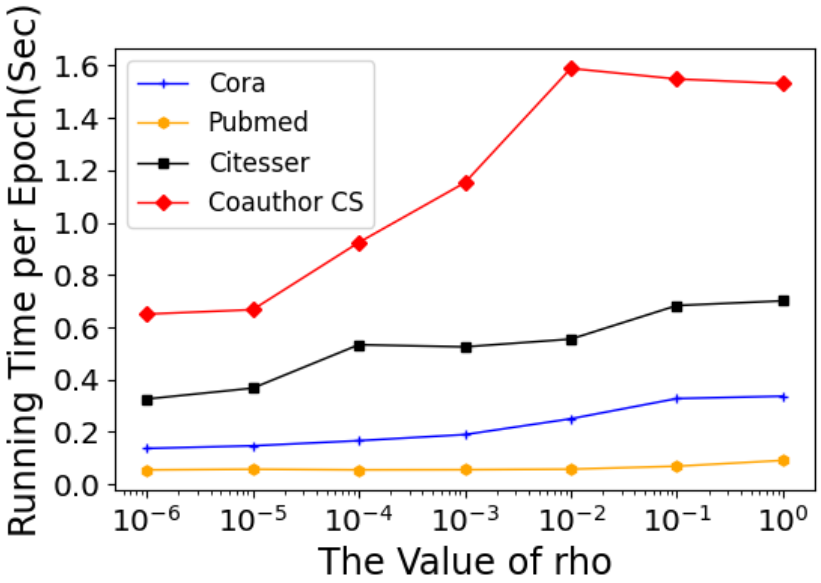
(d). Coauthor CS.

Figure: Test accuracy of all methods: the mDLAM algorithm performs the best.

Running Time Analysis



(a). Running time versus the number of hidden units.



(b). Running time versus the value of ρ.

Figure: The relationship between the running time and: (a) the number of hidden units; (b) the value of ρ: the running time increases linearly with them in general.

The effect of ρ

Cora					
Epoch	40	80	120	160	200
$\rho = 1 \times 10^{-4}$	0.677	0.695	0.695	0.693	0.692
$\rho = 1 \times 10^{-3}$	0.664	0.701	0.721	0.737	0.742
$\rho = 1 \times 10^{-2}$	0.562	0.581	0.604	0.623	0.638
Pubmed					
Epoch	40	80	120	160	200
$\rho = 1 \times 10^{-4}$	0.471	0.407	0.407	0.407	0.407
$\rho = 1 \times 10^{-3}$	0.663	0.645	0.640	0.650	0.649
$\rho = 1 \times 10^{-2}$	0.743	0.758	0.762	0.768	0.773
Citeseer					
Epoch	40	80	120	160	200
$\rho = 1 \times 10^{-4}$	0.528	0.529	0.530	0.531	0.535
$\rho = 1 \times 10^{-3}$	0.651	0.665	0.664	0.664	0.666
$\rho = 1 \times 10^{-2}$	0.631	0.638	0.642	0.648	0.653
Coauthor CS					
Epoch	40	80	120	160	200
$\rho = 1 \times 10^{-4}$	0.843	0.881	0.888	0.896	0.894
$\rho = 1 \times 10^{-3}$	0.780	0.807	0.825	0.839	0.835
$\rho = 1 \times 10^{-2}$	0.688	0.719	0.724	0.737	0.738

Table: The effect of ρ on four datasets: it affects performance significantly.

The effect of ε

Cora					
Epoch	40	80	120	160	200
$\varepsilon^0 = 1$	0.620	0.679	0.712	0.735	0.743
$\varepsilon^0 = 10$	0.646	0.689	0.718	0.741	0.741
$\varepsilon^0 = 100$	0.664	0.701	0.721	0.737	0.742
Pubmed					
Epoch	40	80	120	160	200
$\varepsilon^0 = 1$	0.717	0.744	0.756	0.759	0.763
$\varepsilon^0 = 10$	0.731	0.753	0.759	0.762	0.765
$\varepsilon^0 = 100$	0.743	0.758	0.762	0.768	0.773
Citeseer					
Epoch	40	80	120	160	200
$\varepsilon^0 = 1$	0.564	0.615	0.638	0.653	0.663
$\varepsilon^0 = 10$	0.584	0.626	0.643	0.657	0.662
$\varepsilon^0 = 100$	0.640	0.656	0.664	0.663	0.668
Coauthor CS					
Epoch	40	80	120	160	200
$\varepsilon^0 = 1$	0.834	0.875	0.887	0.893	0.894
$\varepsilon^0 = 10$	0.852	0.866	0.892	0.893	0.893
$\varepsilon^0 = 100$	0.843	0.881	0.888	0.896	0.894

Table: The effect of the initial value of ε on four datasets: it only affects the convergence speed, but have little effect on final performance.

Table of Contents

Conclusion

In this talk,

- We propose a novel formulation of neural networking training via inequality approximation.
- We present an efficient mDLAM algorithm with Nesterov acceleration.
- We investigate the linear convergence of the proposed mDLAM algorithm under mild conditions.
- Extensive experiments have been conducted to demonstrate the convergence, effectiveness, efficiency nd robustness of the proposed mDLAM algorithm.

Code

Our code is available at `https://github.com/xianggebenben/mDLAM`. Please cite our following paper if you use our MLP code in your own work: Wang, Junxiang, Hongyi Li, and Liang Zhao. "Accelerated gradient-free neural network training by multi-convex alternating optimization." Neurocomputing 487 (2022): 130-143.

Reference

Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein.
Training neural networks without gradients: A scalable admm approach.
In *International Conference on Machine Learning*, pages 2722–2731, 2016.

Junxiang Wang, Fuxun Yu, Xiang Chen, and Liang Zhao.
Admm for efficient deep learning with global convergence.
In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 111–119, 2019.