Scalable ADMM for Convex Neural Networks on GPU

Miria Feng, Zachary Shah, Daniel Abraham

EE364B, Electrical Engineering, Stanford University

Abstract

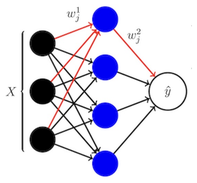
We aim to better understand the fundamental success of Neural Networks, and leverage the provable guarantees of convex optimization within a deep learning framework. Prior work has shown that training a ReLU ([2]) two-layer neural network can be reformulated as group-lasso optimization problem. **Our project investigates:**

- ADMM-style methods for solving the convex equivalent two-layer ReLU problem.
- Effect of the Nyström-ADMM Preconditioner[4] on the problem's condition number and robustness.
- Comparison of GPU accelerated methods versus CPU based methods.

ADMM Approach: ReLU Convex Formulation

Non-convex formulation of a two-layer ReLU-MLP with weights $w_j^{(1)}$, $w_j^{(2)}$:

$$f_{\text{NCVX-MLP}}(x) = \sum_{j=1}^{m} \left(x^T w_j^{(1)} \right)_+ w_j^{(2)} \qquad (1)$$



- $X \in \mathbb{R}^{n \times d}$ is a two layer ReLU-MLP, usually trained in a stochastic setting.
- Scalable in large datasets, but sensitive to hyperparameter tuning and lacks optimality guaranttees.

Equivalent convex reformulation ([3]):

$$f_{\text{CVX-MLP}}(x) = \sum_{i=1}^{P_s} D_i x^T (u_j - v_j) \qquad (2)$$

- $D_i = \mathbf{diag}(\mathbb{I}[X^T h_i \geq 0])$ are diagonal matrices used to sample P_s activation patterns of the ReLU network.
- Recent work [1] applied ADMM by introducing slack variables to arrive at a convex reformulation with mean squared error loss:

$$\min_{v,s,u} ||Fu - y||_2^2 + \beta ||v||_{2,1} + \mathbb{I}_{\geq 0}(s) \quad \text{s.t.} \quad u = v, \; Gu = s \qquad (3)$$

- Matrix $F \in \mathbb{R}^{n \times 2dP_s}$ in this formulation is block-wise constructed by $D_i X$ terms. This optimization problem yields the (simplified) ADMM updates explicitly:

$$\textbf{Primal u update}: \quad Au^{k+1} = b \quad \text{for } A = I + \frac{1}{\rho} F^T F + G^T G \qquad (4a)$$

$$\textbf{Primal v update}: \quad v^{k+1} = \mathbf{prox}_{\frac{\beta}{\rho}||\cdot||_2}(u^{k+1} + \lambda^k) \qquad (4b)$$

$$\textbf{Dual update}: \quad \lambda^{k+1} = \lambda^k + \gamma_\alpha (u^{k+1} - v^{k+1}) \qquad (4c)$$

Methods and Experiments

The problem: (4a) involves a $(2dP_s \times 2dP_s)$ linear system solve, which typically invokes an expensive Cholesky decomposition. We experiment with several approximate iterative methods to solve this in a scalable and accelerated way:

1. Randomized block coordinate descent (RBCD): Low memory compute with fast convergence.
2. Conjugate gradient (CG): Simple Baseline.
3. Nystrom-preconditioned Conjugate gradient (nysPCG): Fast convergence as a result of the "nice" condition number.

We also conduct JAX and PyTorch acceleration to benchmark against CPU performance.

Main Results

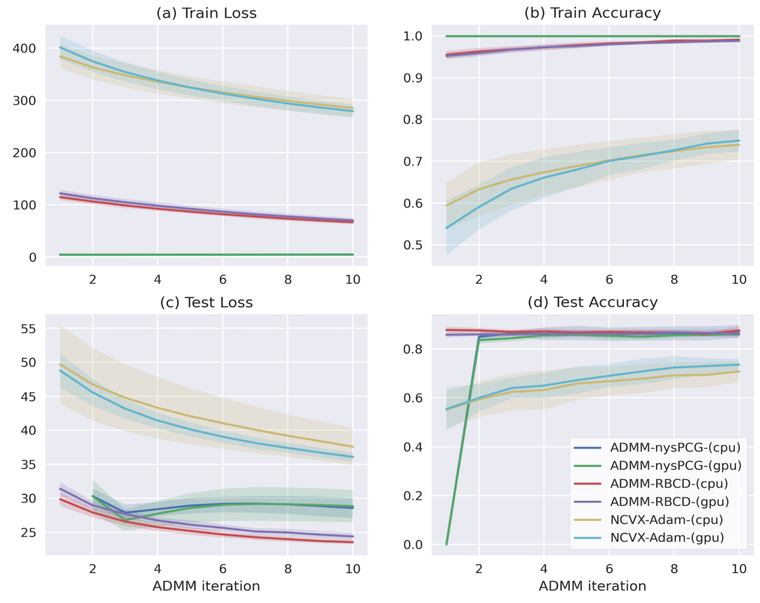


Figure 1. Inexact ADMM Performance on CIFAR-10

- Figure (1) show the performance of our solvers on CIFAR-10 ($n = 10000$, $d = 3072$, $P_s = 50$) after 10 ADMM steps, benchmarked against a two-layer ReLU network trained with non-convex backprop (Adam optimizer with learning rate 10^{-4}).
- Figure (3) show the corresponding solve times for these results.

GPU and Condition Numbers

- Figure (2) shows initial results of preconditioners on the condition number κ.
- Nystrom Sketch Preconditioner had the best effect and increased robustness against hyperparameter tuning!

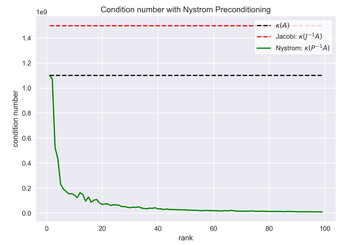


Figure 2. Condition number vs. Rank(A)

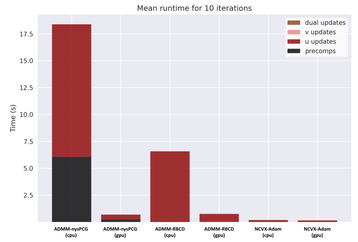


Figure 3. Solve time breakdowns.

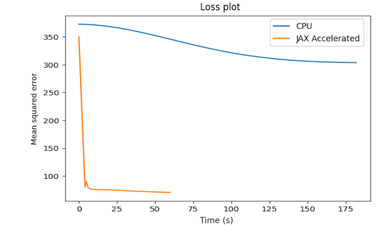


Figure 4. JAX on RBCD method.

- Figure (4): shows the first algorithm, RBCD, to leverage JAX acceleration on A100 GPU with Just In Time compilation against standard CPU usage.
- So far: RBCD methods, initial point methods, and truncated newton show better efficiency in JAX but more experiments needed

Conclusions and Next Steps

- Promising results of Nyström Preconditioning can be applied to RBCD, which has the best performance thus far.
- Package all methods with JAX acceleration backend for easy pip install.
- Provide provable guarantees on networks satisfying desirable input-output properties or specifications.
- Code at: `https://github.com/zachary-shah/admmNN`

References

[1] Yatong Bai, Tanmay Gautam, and Somayeh Sojoudi. Efficient global optimization of two-layer ReLU networks: Quadratic-time algorithms and adversarial training. January 2022.

[2] Aaron Mishkin, Arda Sahiner, and Mert Pilanci. Fast convex optimization for two-layer relu networks: Equivalent model classes and cone decompositions, 2022.

[3] Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7695–7705. PMLR, 13–18 Jul 2020.

[4] Shipu Zhao, Zachary Frangella, and Madeleine Udell. Nysadmm: faster composite convex optimization via low-rank approximation. In *International Conference on Machine Learning*, pages 26824–26840. PMLR, 2022.