ALPS: Improved Optimization for Highly Sparse One-Shot Pruning for Large Language Models

NeurIPS 2024

Xiang Meng, MIT Operations Research Center
Kayhan Behdin, MIT Operations Research Center
Haoyue Wang, MIT Operations Research Center
Rahul Mazumder, MIT Sloan School of Management

NEURAL INFORMATION PROCESSING SYSTEMS

Large Language Models Compression via Unstructured Pruning

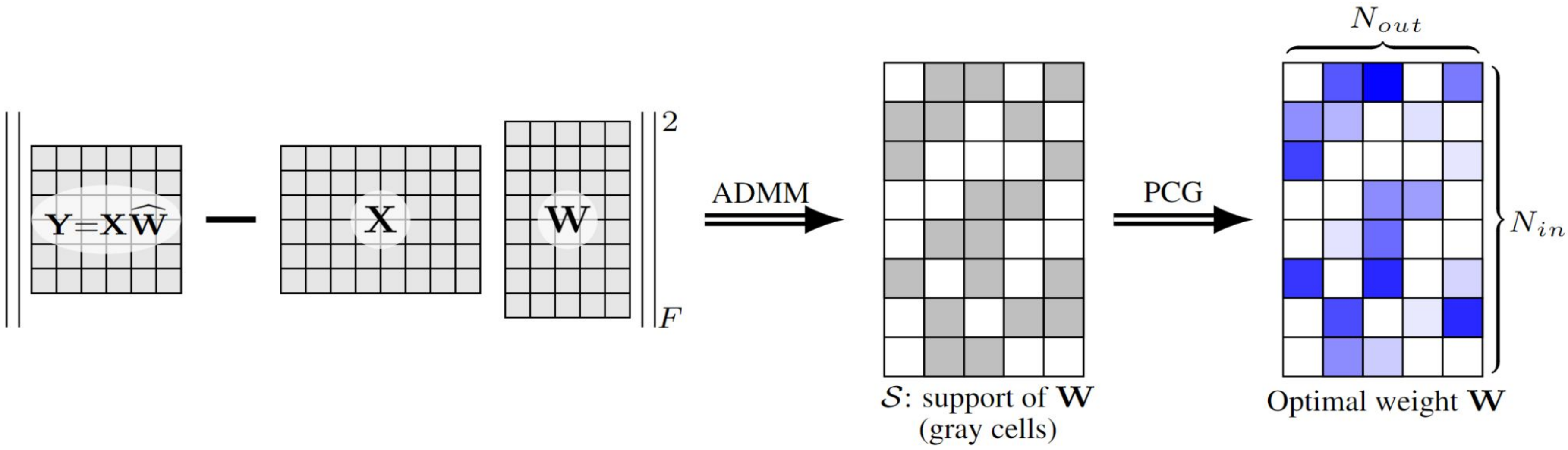
- ❖ Large language models like ChatGPT revolutionize our daily life with massive scale. However, with up to hundreds of billions of parameters, they pose significant challenges in storage and inference

- ❖ **Network Pruning**: reduces model size by identifying redundant weights and setting them to zero

- ❖ Focus on **one-shot unstructured pruning**: compress a pre-trained model once using limited data, without fine-tuning

- ❖ Challenges:
  - ➢ CNN pruning methods don't scale to LLMs;
  - ➢ current LLM pruning relies on heuristics, potentially leading to suboptimal compression-accuracy trade-offs

Overview of ALPS — ADMM-based LLM Pruning in one-Shot



- Layerwise reconstruction objective with an ℓ_0 constraint on the weights.
- ADMM is employed to determine high-quality support for the weight matrix $\mathbf{W}$.
- Restrict the optimization problem to support obtained by ADMM, apply a modified PCG to compute optimal weights within the support.

ADMM for Layer-wise Unstructured Pruning

- Layer-wise pruning problem

$$\min_{\mathbf{W}} \; Q(\mathbf{W}) := \|\mathbf{X}\widehat{\mathbf{W}} - \mathbf{X}\mathbf{W}\|_F^2 + \lambda_2\|\widehat{\mathbf{W}} - \mathbf{W}\|_F^2 \quad \text{s.t.} \; \|\mathbf{W}\|_0 \le k$$

- Reformulation via operater splitting

$$\min_{\mathbf{W},\mathbf{D}} \; Q(\mathbf{W}) + \infty \cdot \mathbf{1}_{\|\mathbf{D}\|_0 > k} \quad \text{s.t.} \quad \mathbf{W} = \mathbf{D}$$

- Augmented Lagrangian

$$L_\rho(\mathbf{W}, \mathbf{D}, \mathbf{V}) = Q(\mathbf{W}) + \infty \cdot \mathbf{1}_{\|\mathbf{D}\|_0 > k} + \langle \mathbf{V}, \mathbf{W} - \mathbf{D} \rangle + \frac{\rho}{2}\|\mathbf{W} - \mathbf{D}\|_F^2$$

- ADMM updates

$$\begin{aligned}
\mathbf{W}^{(t+1)} &= \arg\min_{\mathbf{W}} L_\rho(\mathbf{W}, \mathbf{D}^{(t)}, \mathbf{V}^{(t)}) = \left(\mathbf{X}^\top\mathbf{X} + (\lambda_2 + \rho)\mathbf{I}\right)^{-1}\left((\mathbf{X}^\top\mathbf{X} + \lambda_2\mathbf{I})\widehat{\mathbf{W}} - \mathbf{V}^{(t)} + \rho\mathbf{D}^{(t)}\right) \\
\mathbf{D}^{(t+1)} &= \arg\min_{\mathbf{D}} L_\rho(\mathbf{W}^{(t+1)}, \mathbf{D}, \mathbf{V}^{(t)}) = P_k\left(\mathbf{W}^{(t+1)} + \mathbf{V}^{(t)}/\rho\right), \\
\mathbf{V}^{(t+1)} &= \mathbf{V}^{(t)} + \rho(\mathbf{W}^{(t+1)} - \mathbf{D}^{(t+1)}),
\end{aligned}$$

ADMM for Layer-wise Unstructured Pruning

$$\mathbf{W}^{(t+1)} = \arg\min_{\mathbf{W}} L_\rho(\mathbf{W}, \mathbf{D}^{(t)}, \mathbf{V}^{(t)}) = \left(\mathbf{X}^\top\mathbf{X} + (\lambda_2 + \rho)\mathbf{I}\right)^{-1}\left((\mathbf{X}^\top\mathbf{X} + \lambda_2\mathbf{I})\widehat{\mathbf{W}} - \mathbf{V}^{(t)} + \rho\mathbf{D}^{(t)}\right)$$
$$\mathbf{D}^{(t+1)} = \arg\min_{\mathbf{D}} L_\rho(\mathbf{W}^{(t+1)}, \mathbf{D}, \mathbf{V}^{(t)}) = P_k\left(\mathbf{W}^{(t+1)} + \mathbf{V}^{(t)}/\rho\right),$$
$$\mathbf{V}^{(t+1)} = \mathbf{V}^{(t)} + \rho(\mathbf{W}^{(t+1)} - \mathbf{D}^{(t+1)}),$$

All updates can be efficiently computed via torch (on GPU)

Increase ρ along iterations; ensures convergence & finding high quality solutions

The rate of change of the support between consecutive iterations, comparing ALPS with ADMM using a fixed ρ

Supp change / Iter	5	10	20	30	50	100
ALPS	20.2%	17.0%	2.8%	0.0%	0.0%	0.0%
ADMM($\rho = 0.3$)	6.4%	7.0%	7.0%	7.0%	6.9%	6.9%
ADMM($\rho = 3$)	0.2%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%

Efficiently Refining Weights after Support Stabilization

Post-processing: refining the solution within support given by ADMM

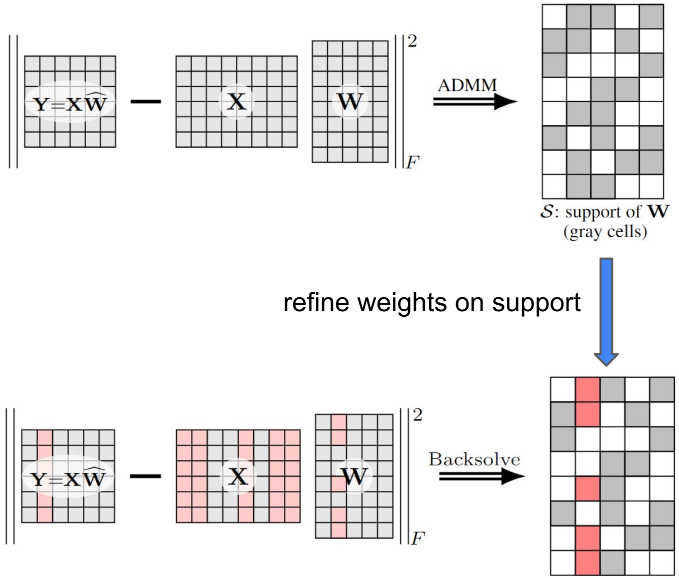
$$\min_{\mathbf{W}} \ Q(\mathbf{W}) \quad \text{s.t.} \ \ \mathrm{Supp}(\mathbf{W}) \subset \mathcal{S}.$$

Decomposes into separate least squares problems across the columns of W.

Direct backsolve — solving >10k different linear systems (very costly)

Proposed: modified conjugate gradient method solving **all columns simultaneously**

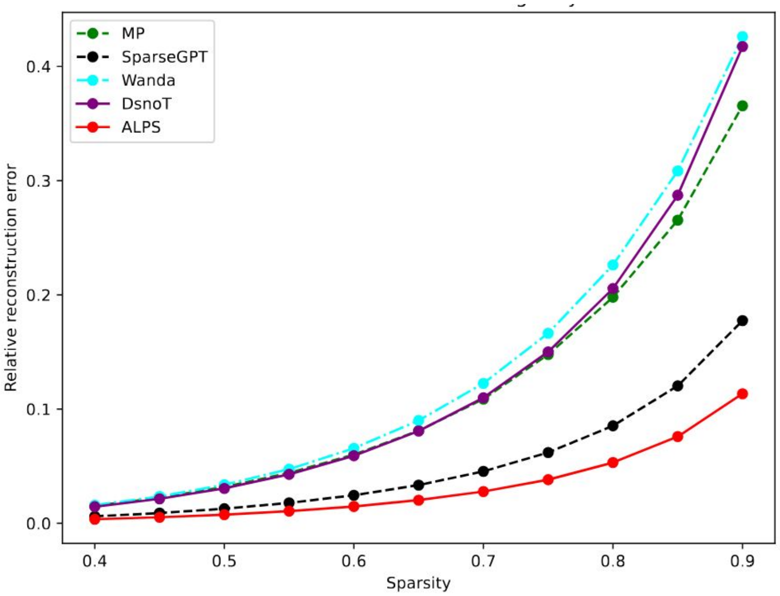
Exploit GPU **parallelism**, >100x acceleration (table)

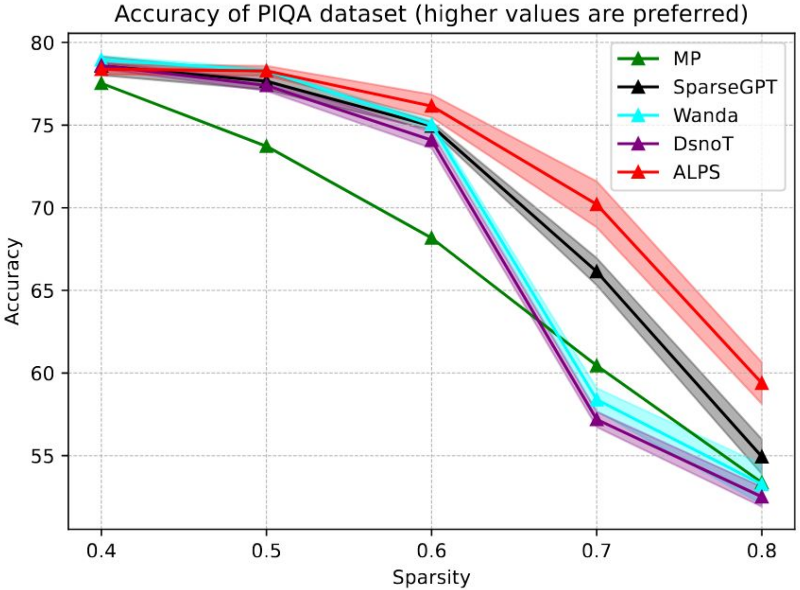


Sparsity	*ALPS*		Backsolve	
	Time(s)	Error	Time(s)	Error
0.5	0.77	1.24e-2	131	1.10e-2
0.6	0.79	2.43e-2	95.0	2.25e-2
0.7	0.78	4.56e-2	64.1	4.38e-2

Experimental Results

Comparing with (i)(MP, [Han et al., 2015]), (ii) SparseGPT [Frantar and Alistarh, 2023], (iii) Wanda [Sun et al. 2023], and (iv) DSnoT [Zhang et al., 2023]





Pruning results on LLaMA-7B. ALPS achieves much lower reconstruction objective (left), which translate to higher performance in various tasks (right).