How to compress neural networks

Vladimír Boža

November 26, 2025

LLMs - what are we dealing with

Llama2-70B:

```
(embed_tokens): Embedding(32000, 8192)
80 x LlamaDecoderLayer(
  (self_attn): LlamaAttention(
    (q_proj): Linear(8192, 8192)
    (k_proj): Linear(8192, 1024)
    (v_proj): Linear(8192, 1024)
    (o_proj): Linear(8192, 8192)
  )
  (mlp): LlamaMLP(
    (gate_proj): Linear(8192, 28672)
    (up_proj): Linear(8192, 28672)
    (down_proj): Linear(28672, 8192)
  )
)
(lm_head): Linear(8192, 32000)*
```

*Some nonparametric parts left out for clarity:
`LlamaRotaryEmbedding()`, `SiLUActivation()`, `LlamaRMSNorm()`

LLMs - what are we dealing with

Llama2-70B:

```
(embed_tokens): Embedding(32000, 8192)
80 x LlamaDecoderLayer(
  (self_attn): LlamaAttention(
    (q_proj): Linear(8192, 8192)
    (k_proj): Linear(8192, 1024)
    (v_proj): Linear(8192, 1024)
    (o_proj): Linear(8192, 8192)
  )
  (mlp): LlamaMLP(
    (gate_proj): Linear(8192, 28672)
    (up_proj): Linear(8192, 28672)
    (down_proj): Linear(28672, 8192)
  )
)
(lm_head): Linear(8192, 32000)*
```

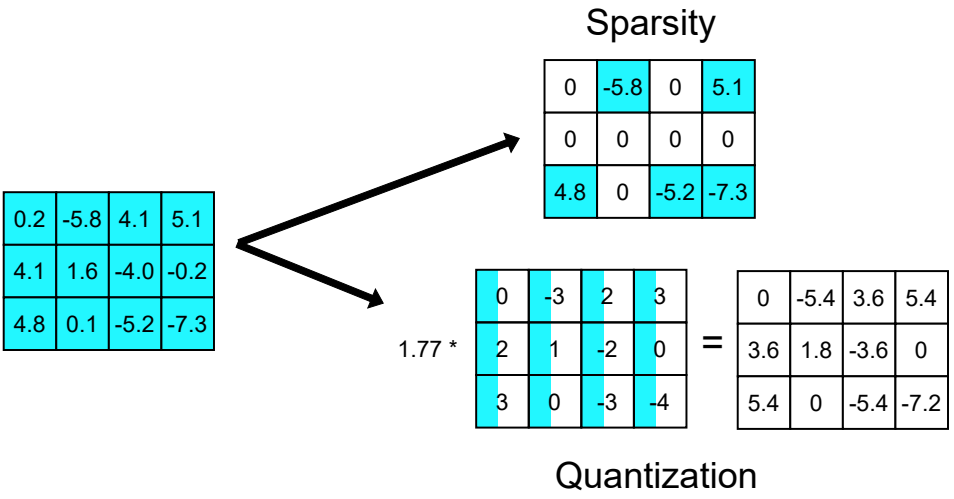
*Some nonparametric parts left out for clarity:
`LlamaRotaryEmbedding()`, `SiLUActivation()`, `LlamaRMSNorm()`

- 70b Float16 parameters
- 140GB of memory
- 99% of it is matrix multiplication
- HW needs for inference:
  - **2x 80-GB A100 GPUs**
  - **4x 40-GB A100 GPUs**
  - **2x 80-GB H100 GPUs**

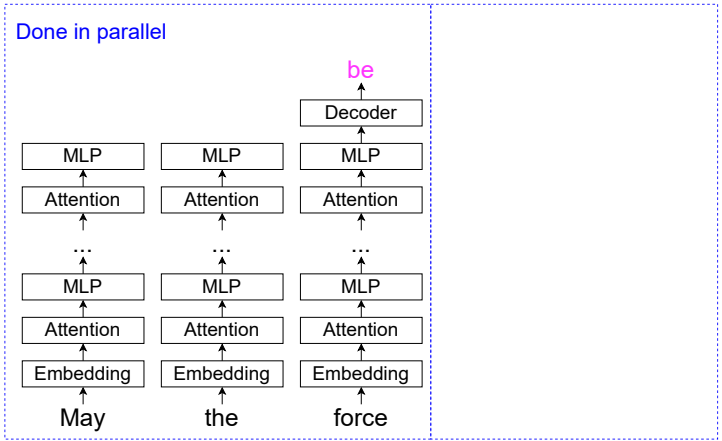
$50k to buy

$3k/month to rent

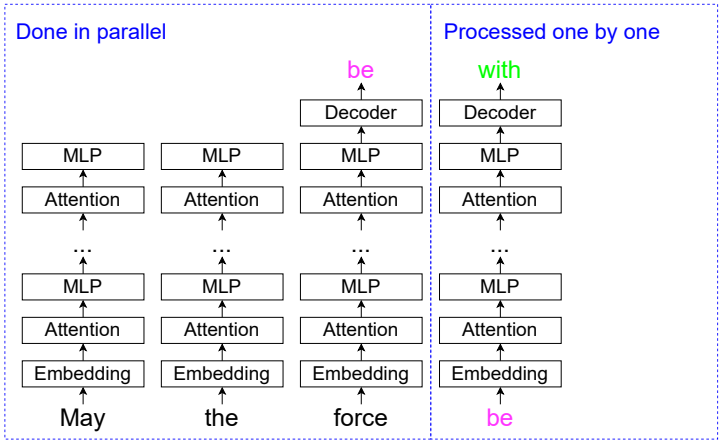
How to make LLMs smaller



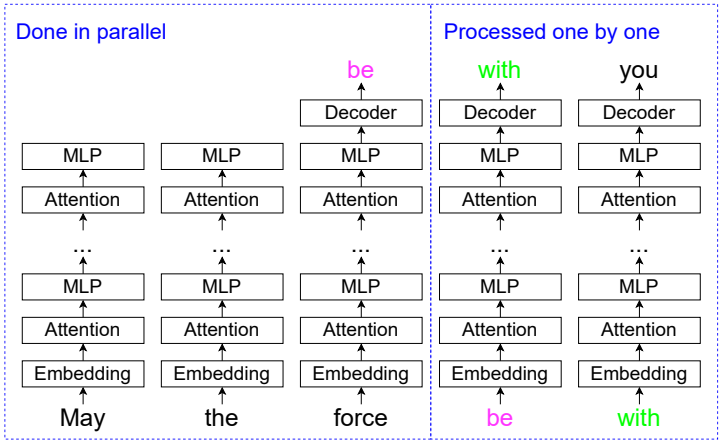
LLM inference



LLM inference



Smaller means faster for local LLMs



For each predicted token, we need to load all of the weights

Smaller weights $\rightarrow$ faster decoding

Layer-wise pruning

- Hard to tune the whole model at once
- Optimize each layer separately
- Capture calibration input X for each layer
- Solve:

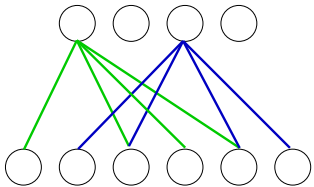
$$\min_{W_p} ||XW - XW_p||_2^2$$

$$\text{s.t. } ||W_p||_0 \leq S$$

Somebody tells me sparsity mask M

$\min_{W_p} ||XW - XW_p||_2^2$

s.t. $W_p \odot (1 - M) = 0$



W has shape $n \times m$

m independent linear regressions $((X^T X)^{-1} X^T y)$: $O(mn^3)$ time.

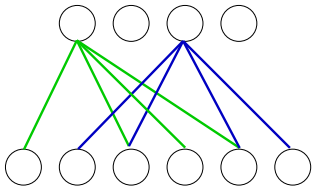
Gradient descent $(X^T(XW - XW_p) \odot M)$: $O(mn^2)$ per iteration, but many iterations.

Can we do better?

Somebody tells me sparsity mask M

$\min_{W_p} ||XW - XW_p||_2^2$

s.t. $W_p \odot (1 - M) = 0$



This is just a constrained optimization problem

Use Alternating Direction Method of Multipliers (ADMM) to solve it

"Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers" by Boyd is a really good reading.

ADMM recap - augmented Lagrangian

$$\text{minimize } f(x)$$
$$\text{subject to } Ax = B$$

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + (\rho/2)||Ax - b||_2^2$$

$$x^{k+1} = \text{argmin}_x \; L_\rho(x, y^k)$$
$$y^{k+1} = y^k + \rho(Ax^{k+1} - b)$$

ADMM recap - problem split

$$\text{minimize } f(x) + g(z)$$
$$\text{subject to } Ax + Bz = c$$

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)||Ax + Bz - c||_2^2$$

$$x^{k+1} = \text{argmin}_x \; L_\rho(x, z^k, y^k)$$
$$z^{k+1} = \text{argmin}_z \; L_\rho(x^{k+1}, z, y^k)$$
$$y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)$$

ADMM recap - substitution

$$\text{minimize } f(x) + g(z)$$
$$\text{subject to } Ax + Bz = c$$

$$u = \frac{1}{\rho} y$$

$$x^{k+1} = \text{argmin}_x \; f(x) + (\rho/2)||Ax + Bz^k - c + u^k||_2^2$$
$$z^{k+1} = \text{argmin}_z \; g(z) + (\rho/2)||Ax^{k+1} + Bz - c + u^k||_2^2$$
$$u^{k+1} = u^k + Ax^{k+1} + Bz^{k+1} - c$$

ADMM for constrained optimization

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } x \in C \end{aligned}$$

ADMM for constrained optimization

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } x \in C \end{aligned}$$

Let $g(z) = 0$ if $z \in C$ otherwise $g(z) = \inf$.

$$\begin{aligned} &\text{minimize } f(x) + g(z) \\ &\text{subject to } x = z \end{aligned}$$

ADMM for constrained optimization

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } x \in C \end{aligned}$$

Let $g(z) = 0$ if $z \in C$ otherwise $g(z) = \inf$.

$$\begin{aligned} &\text{minimize } f(x) + g(z) \\ &\text{subject to } x = z \end{aligned}$$

$$\begin{aligned} x^{k+1} &= \text{argmin}_x \; f(x) + (\rho/2)||x - z^k + u^k||_2^2 \\ z^{k+1} &= \text{argmin}_z \; g(z) + (\rho/2)||x^{k+1} - z + u^k||_2^2 \\ u^{k+1} &= u^k + x^{k+1} - z^{k+1} \end{aligned}$$

z-step is just a L2 projection of $x + u$ upon C.

Back to masked regression

$$\min_{W_p} ||XW - XW_p||_2^2$$
$$\text{s.t. } W_p \odot (1 - M) = 0$$

$$W_p^{k+1} = (X^T X + \rho I)^{-1}(X^T XW + \rho(Z^k - U^k))$$
$$Z^{k+1} = M \odot (W_p^{k+1} + U^k)$$
$$U^{k+1} = U^k + W_p^{k+1} - Z^{k+1}$$

Update rule discussion

$$
\begin{aligned}
W_p^{k+1} &= (X^T X + \rho I)^{-1}(X^T X W + \rho(Z^k - U^k)) \\
Z^{k+1} &= M \odot (W_p^{k+1} + U^k) \\
U^{k+1} &= U^k + W_p^{k+1} - Z^{k+1}
\end{aligned}
$$

- Precompute $(X^T X + \rho I)^{-1}$ and $X^T X W$.
- Update complexity same as gradient descent step $O(mn^2)$
- W update is similar to Ridge regression. Here, we pull harder for terms we want to get to zero.

How to find mask

- Prune elements with smallest $W + U$
- Gradually increase sparsity from zero to target one over first p iterations (using cubic schedule)
- Usually we prune during the first 15 iterations out of 20

How to find mask

- Prune elements with smallest $W + U$
- Gradually increase sparsity from zero to target one over first p iterations (using cubic schedule)
- Usually we prune during the first 15 iterations out of 20

$$
\begin{aligned}
W_p^{k+1} &= (X^T X + \rho I)^{-1}(X^T X W + \rho(Z^k - U^k)) \\
M^{k+1} &= \texttt{find\_largest}(\texttt{W}_\texttt{p}^{\texttt{k+1}} + \texttt{U}^\texttt{k}) \\
Z^{k+1} &= M^{k+1} \odot (W_p^{k+1} + U^k) \\
U^{k+1} &= U^k + W_p^{k+1} - Z^{k+1}
\end{aligned}
$$

Results

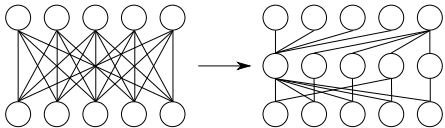
Table: Perplexity of pruned LLaMA-2 variants on WikiText

Method	Sparsity	7B	13 B	70B
Dense	0 %	5.12	4.57	3.12
SparseGPT	50 %	6.51	5.63	3.98
ADMM-Grad	50 %	**6.33**	**5.52**	**3.95**
SparseGPT	60 %	9.58	7.80	4.98
ADMM-Grad	60 %	**8.70**	**7.09**	**4.81**
SparseGPT	2:4	10.17	8.32	5.40
ADMM-Grad	2:4	**9.74**	**7.78**	**5.19**

More in: "Fast and Effective Weight Update for Pruned Large Language Models." TMLR 2024.


MMM. ALWAYS TWO THERE ARE
imgflip.com

What if?



Factorize each weight matrix into two sparse matrices (Double sparse factorization).

$$\min_{A,B} ||W - AB||_2^2 \tag{1}$$

$$\text{s.t. } ||A||_0 + ||B||_0 \leq S \tag{2}$$

Generalizes low rank, monarch, ordinary sparsity (sort of).

Heuristic solution idea

- Run alternating minimization (fix A find B via ADMM, ...)
- In later iterations, reuse solution from previous steps as starting point
- Do now solve problem too hard during first steps

Alternating minimization

Initialize $A^{(0)}, B^{(0)}$

$U_a^{(0)} = 0 \cdot A, U_b^{(0)} = 0 \cdot B$

for $k = 1..n$ **do**

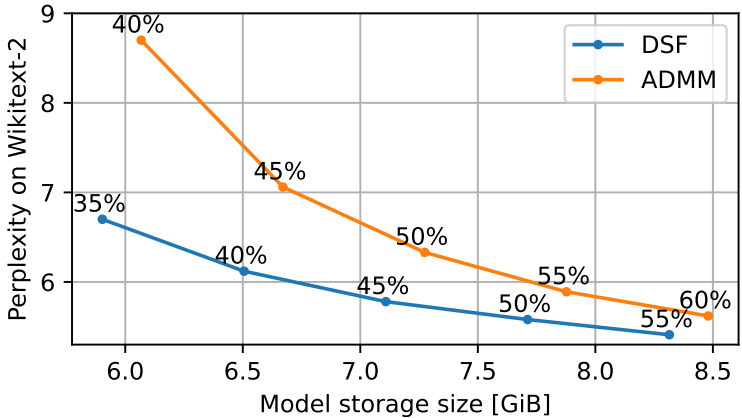
$\rho_0 = \min(1.0, k/(n-3))^3$

$B^{(k)}, U_b^{(k)} \leftarrow$ solve $\arg\min ||AB - W||_F$, st. $||B||_0 \leq z_b$ via m iterations of ADMM with starting point $B^{k-1}, U_b^{(k-1)}$ and starting ρ_0

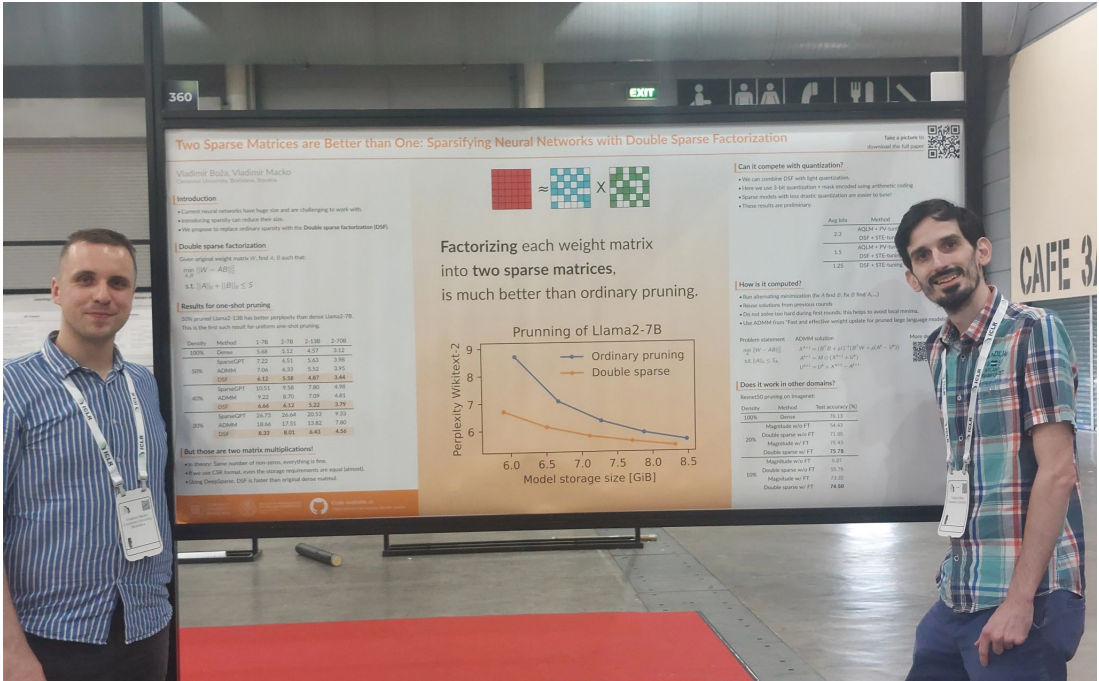
$A^{(k)}, U_a^{(k)} \leftarrow$ solve $\arg\min ||AB - W||_F$, st. $||A||_0 \leq z_a$ via m iterations of ADMM with starting point $A^{k-1}, U_a^{(k-1)}$ and starting ρ_0

end for

Results (16 bits per non-zero, binary mask)



More in: "Two Sparse Matrices are Better than One: Sparsifying Neural Networks with Double Sparse Factorization." ICLR 2025.


360
EXIT
Two Sparse Matrices are Better than One: Sparsifying Neural Networks with Double Sparse Factorization
Take a picture to download the full paper
Vladimír Boža, Vladimír Macko
Comenius University, Bratislava, Slovakia
Introduction
• Current neural networks have huge size and are challenging to work with.
• Introducing sparsity can reduce their size.
• We propose to replace ordinary sparsity with the Double sparse factorization (DSF).
Double sparse factorization
Given original weight matrix W, find A, B such that:
$\min_{A,B} \|W - AB\|_2^2$
s.t. $\|A\|_0 + \|B\|_0 \leq S$
Results for one-shot pruning
30% pruned Llama2-13B has better perplexity than dense Llama2-7B.
This is the first such result for uniform one-shot pruning.
Density | Method | 1-7B | 2-7B | 2-13B | 2-70B
100% | Dense | 5.68 | 5.12 | 4.57 | 3.12
50% | SparseGPT | 7.22 | 6.51 | 5.63 | 3.98
ADMM | 7.06 | 6.33 | 5.52 | 3.95
DSF | 6.12 | 5.58 | 4.87 | 3.44
40% | SparseGPT | 10.51 | 9.58 | 7.80 | 4.98
ADMM | 9.22 | 8.70 | 7.09 | 4.81
DSF | 6.66 | 6.12 | 5.22 | 3.79
30% | SparseGPT | 26.73 | 26.64 | 20.53 | 9.33
ADMM | 18.66 | 17.51 | 13.82 | 7.80
DSF | 8.33 | 8.01 | 6.43 | 4.56
But those are two matrix multiplications!
• In theory: Same number of non-zeros, everything is fine.
• It was not CSR format, even the storage requirements are equal (almost).
• Using DeepSparse, DSF is faster than original dense matmul.
Code available at:
github.com/vlejd/sparse_lib
Factorizing each weight matrix into two sparse matrices, is much better than ordinary pruning.
Prunning of Llama2-7B
Perplexity Wikitext-2
Model storage size [GiB]
Ordinary pruning
Double sparse
Can it compete with quantization?
• We can combine DSF with light quantization.
• Here we use 3-bit quantization + mask encoded using arithmetic coding
• Sparse models with less drastic quantization are easier to tune!
• Those results are preliminary.
Avg bits | Method
2.3 | AQLM + PV-tuning
DSF + STE-tuning
1.5 | AQLM + PV-tuning
DSF + STE-tuning
1.25 | DSF + STE-tuning
How is it computed?
• Run alternating minimization (fix A find B, fix B find A,...)
• Reuse solutions from previous rounds
• Do not solve too hard during first rounds: this helps to avoid local minima.
• Use ADMM from "Fast and effective weight update for pruned large language models"
Problem statement
ADMM solution
More details
Does it work in other domains?
Resnet50 pruning on Imagenet:
Density | Method | Test accuracy [%]
100% | Dense | 76.13
20% | Magnitude w/o FT | 54.43
Double sparse w/o FT | 71.05
Magnitude w/ FT | 75.43
Double sparse w/ FT | 75.78
10% | Magnitude w/o FT | 9.07
Double sparse w/o FT | 55.76
Magnitude w/ FT | 73.32
Double sparse w/ FT | 74.50
CAFE 3A


No flammable goods
No durians
Fine S$5000

Quantization is still better than sparsity

Table: Perplexity of pruned LLaMA-2-7B

Method	Size reduction	Perplexity
Dense	-	5.12
50% ADMM	/2	6.33
50% Double sparse	/2	6.12
8bit naive quantization	/2	5.13
4bit naive quantization	/4	5.71
4bit good quantization	/4	5.17
2bit good quantization	/8	5.86

Silver lining: Very good quantization methods are impossible to finetune.

Combine Double sparsity with quantization?

Works, but still limited HW support.

Different extreme idea: Factorize into binary (± 1) matrices.

Almost binary neural networks

Turning a matrix into binary is quite drastic

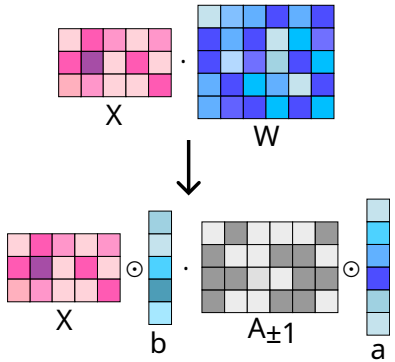
Add scaling factors for each column and row

We want: $W \approx A_{\pm 1} \odot (ab^T)$

Solution:
$A_{\pm 1} = \text{sign}(W)$
a, b are rank-1 decomposition of $|W|$

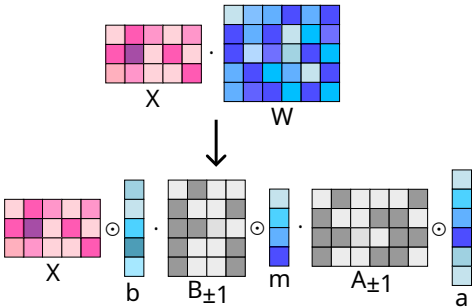
We will call this SVID

Good news: Multiplication by ± 1 matrix can be implemented using additions.



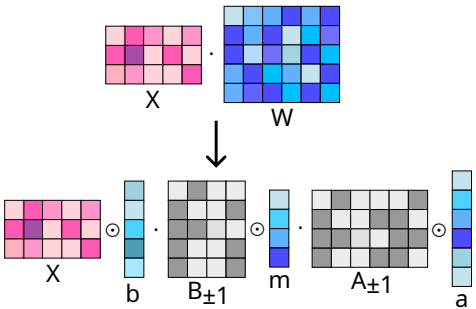
Xu, Yuzhuang, et al. "Onebit: Towards extremely low-bit large language models." NeurIPS 2024

Double binary factorization (DBF)



$$W \approx D_b B_{\pm 1} D_m A_{\pm 1} D_a$$

Double binary factorization (DBF)



$$W \approx D_b B_{\pm 1} D_m A_{\pm 1} D_a$$

Replace most of multiplications with additions

Varying middle dimension determines compression ratio

Can be finetuned if you are brave enough

Finding DBF

We want: $W \approx (D_b B_{\pm 1} D_{m_1})(D_{m_2} A_{\pm 1} D_a) = BA$

Fix B, find A, ... How to find A? ADMM!

Finding DBF

We want: $W \approx (D_b B_{\pm 1} D_{m_1})(D_{m_2} A_{\pm 1} D_a) = BA$

Fix B, find A, ... How to find A? ADMM!

$$\min_A \ ||W - BA||_2^2$$
$$\text{s.t. } A = D_{m_2} A_{\pm 1} D_a$$

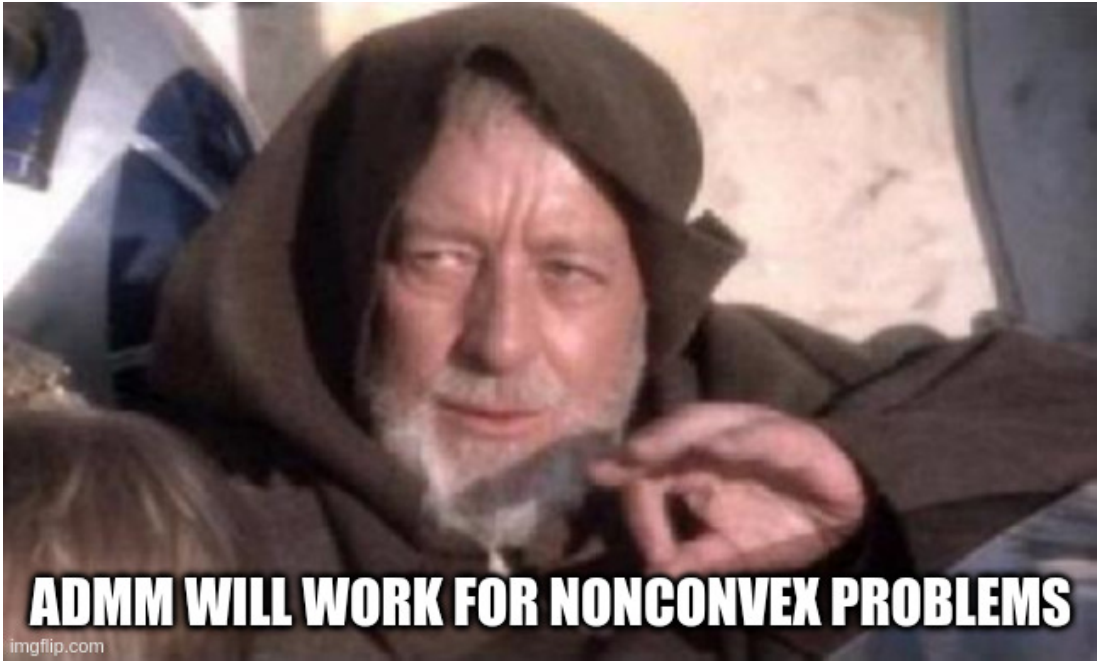
Finding DBF

We want: $W \approx (D_b B_{\pm 1} D_{m_1})(D_{m_2} A_{\pm 1} D_a) = BA$

Fix B, find A, ... How to find A? ADMM!

$$\begin{aligned} &\min_A \ ||W - BA||_2^2 \\ &\text{s.t. } A = D_{m_2} A_{\pm 1} D_a \end{aligned}$$

$$\begin{aligned} X^{k+1} &= (B^T B + \rho I)^{-1}(B^T W + \rho(A^k - U^k)) \\ A_{\pm}1 &= \texttt{sign}(X^{k+1}) \\ D_{m_2}, D_a &= \texttt{rank-1}(|X^{k+1}|) \\ A^{k+1} &= D_{m_2} A_{\pm 1} D_a \\ U^{k+1} &= U^k + X^{k+1} - A^{k+1} \end{aligned}$$


ADMM WILL WORK FOR NONCONVEX PROBLEMS
imgflip.com

DBF results

Table: Results for Llama2-7B.

Avg. bits	Method	Wikitext ppl.
16	Dense	5.12
2	QTIP	5.86
2	DBF	6.09
1	OneBit	9.73
1	DBF	8.76

Table: Results for Llama3-8B.

Avg. bits	Method	Wikitext ppl.
16	Dense	5.54
2	QTIP	7.33
2	DBF	7.30
1.1	BiLLM	28.80
1	DBF	13.57

DBF results

Table: Results for Llama2-7B.

Avg. bits	Method	Wikitext ppl.
16	Dense	5.12
2	QTIP	5.86
2	DBF	6.09
1	OneBit	9.73
1	DBF	8.76

Table: Results for Llama3-8B.

Avg. bits	Method	Wikitext ppl.
16	Dense	5.54
2	QTIP	7.33
2	DBF	7.30
1.1	BiLLM	28.80
1	DBF	13.57

Table: Batch size 1 decoding throughput on Nvidia RTX 4090 for two versions of Llama models.

	Avg. bits	2-7B	3-8b
Original	16	68 tok/s	60 tok/s
DBF	2	153 tok/s (x2.25)	133 tok/s (x2.22)
DBF	1	170 tok/s (x2.50)	174 tok/s (x2.90)

More in: "Addition is almost all you need: Compressing neural networks with double binary factorization."


Addition is almost all you need:
Compressing neural networks with
Double Binary Factorization
ITODLeP-FM workshop, ICML 2025
Abstract
Double Binary Factorization
Finding DBF
Throughput RTX 4090
Results Llama-2-7b
Results Llama-3-8b


INTERNET ARCHIVE
CANADA

Thank you for the attention