Efficient Computation of Low-Rank Representations to Reduce Memory Requirements in Deep Learning

September 11, 2024 | Dr. Carolin Penke | Jülich Supercomputing Centre

ABOUT ME

Technomathe-matics
B.Sc., M.Sc.

Numerical Linear Algebra
Dr. rer. nat.

Large Language Models
Postdoc (Research Software Engineer)

2010 - 2016

2017 - 2021

2022 - 2025

MAX PLANCK INSTITUTE FOR DYNAMICS OF COMPLEX TECHNICAL SYSTEMS MAGDEBURG

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

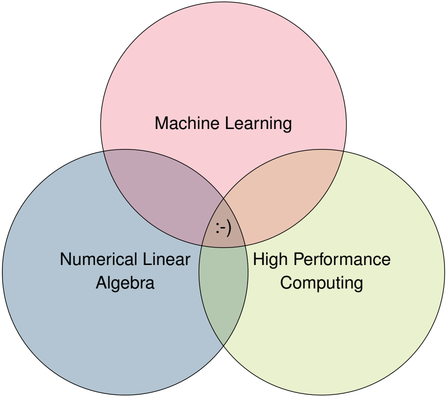
NII Inter-University Research Institute Corporation / Research Organization of Information and Systems
National Institute of Informatics

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

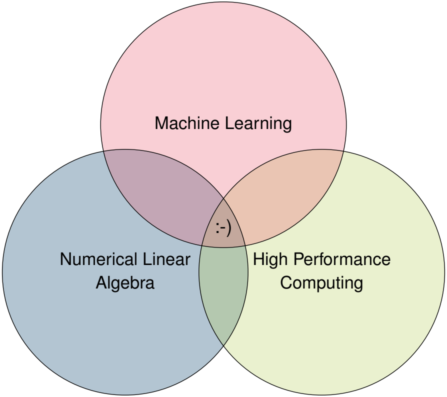
openGPT-X

MPCDF MAX PLANCK COMPUTING & DATA FACILITY

MY RESEARCH INTERESTS



MY RESEARCH INTERESTS



+ Diversity, Equity and Inclusion

WHPC WOMEN IN HPC

JÜLICH Forschungszentrum

WHAT IS A LANGUAGE MODEL?

Currently, huge efforts are directed to training *large language models*!

- Sequence of words $w_1, w_2, \ldots, \in V$ (*Vocabulary*)
- A languange model approximates a probability distribution

$$P(w_t | w_{1:(t-1)})$$

"How likely is a specific word to follow a given sequence of words?"

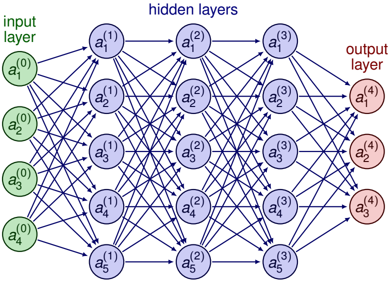
$\rightarrow$ Can be used to generate new texts.

$\rightarrow$ Probabilty of entire sentence:

$$P(w_{1:n}) = \prod_{i=1}^{n} P(w_i | w_{1:i-1})$$

DEEP LEARNING

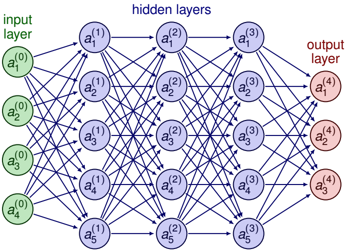
Goal: Learn input-output relations from data



Source: Izaak Neutelings `https://tikz.net/neural_networks/`

- → Forward propagation: Compute activations $a_i^{(j)}$ and loss
- ← Backward propagation: Update weights to minimize loss (gradient descent)
- Repeat until convergence

DEEP LEARNING ARCHITECTURES

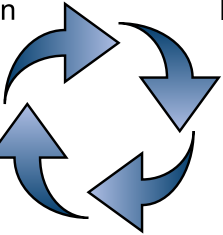


Fully connected neural network, also called feed-forward layer (FFL) or multi-level-perceptron (MLP)

+ Matrix multiplications in forward & backward propagation
  → well-suited for HPC
– No fit for sequential nature of language

Specialized architectures are needed!

HOW NEW ARCHITECTURES EMERGE



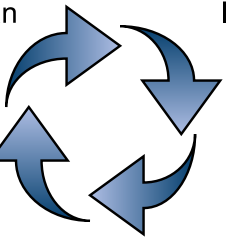
Somebody has "an intuition"

It turns out to work well

Widespread adoption

Further research on how it works

HOW NEW ARCHITECTURES EMERGE



Discomfort for mathematicians

Relationship intuition $\leftrightarrow$ reality: questionable

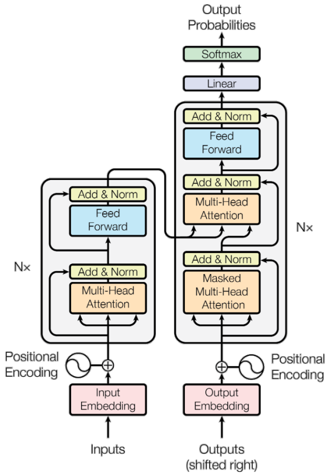
THE TRANSFORMER ARCHITECTURE

The T in GPT

- The transformer architecture was introduced in 2017.
- The main innovation is the attention mechanism

$$\text{SelfAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V \ .$$

- Softmax applied on rows, including masking.
- Q, K, V contain learned representations of input tokens.

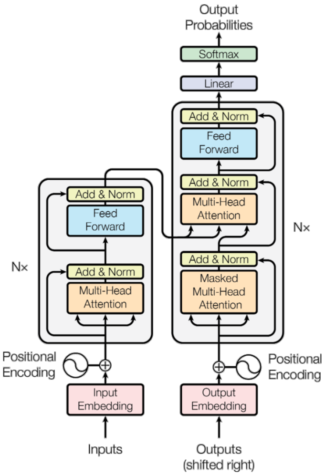


Attention is all you need, A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, 2017

THE TRANSFORMER ARCHITECTURE

The T in GPT

- A transformer neural network stacks a number of transformer layers, each containing an attention block and a feed forward layer.
- Remarkable abilities are shown by large models with many parameters.
- GPT-4: 1.76 trillion parameters (estimated)



Attention is all you need, A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, 2017

TRAINING LARGE MODELS

Training these large models needs

- Lots of computational resources (GPUs!),
- Lots of data.

Pretraining happens on supercomputers.

(R-U. Limbach / Forschungszentrum Jülich)

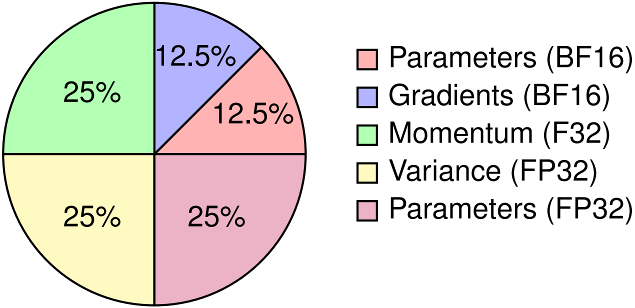
Finetuning of smaller models happens on workstations.

NVIDIA

In both settings, you want to use limited resources efficiently.

GPU MEMORY REQUIREMENTS DURING TRAINING

Using the mixed-precision Adam optimizer.



+ Activations, depending on sequence length and batch size.
- Activations can be reduced using activation checkpointing.

OBSERVING LOW RANK

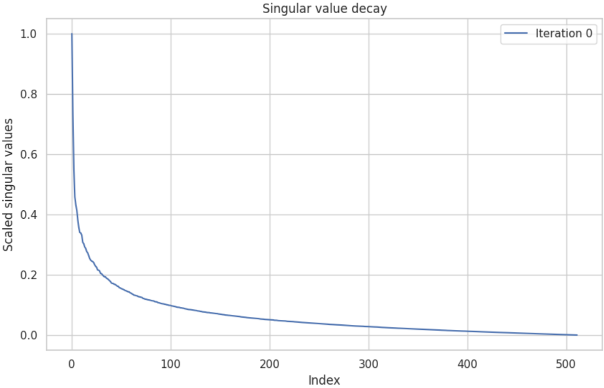


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

OBSERVING LOW RANK

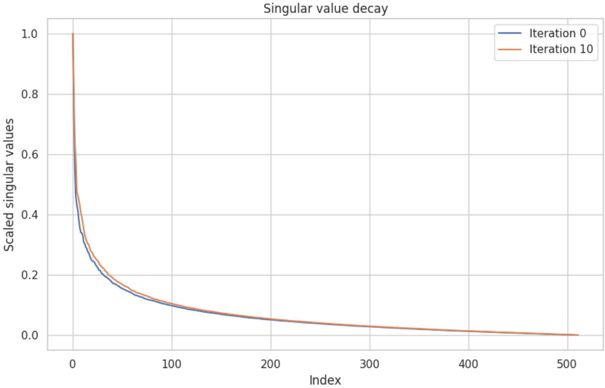


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

OBSERVING LOW RANK

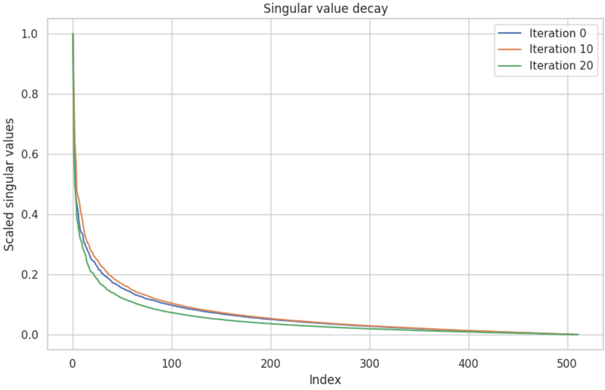


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

OBSERVING LOW RANK

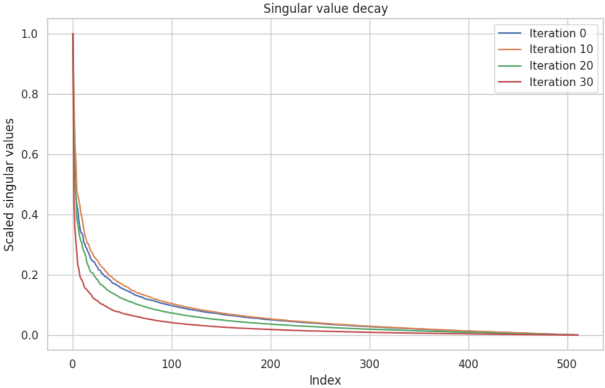


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

OBSERVING LOW RANK

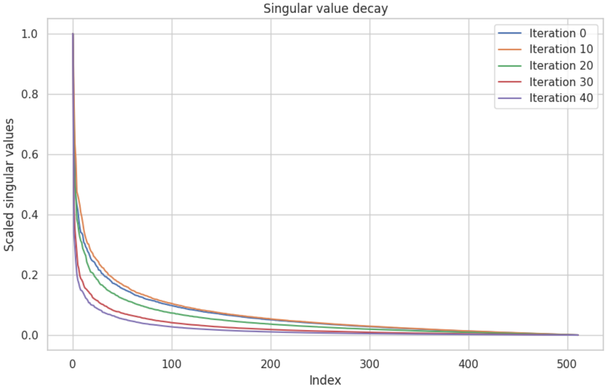


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

OBSERVING LOW RANK

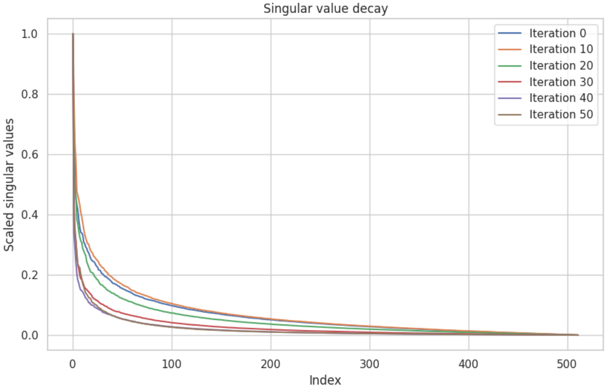


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

OBSERVING LOW RANK

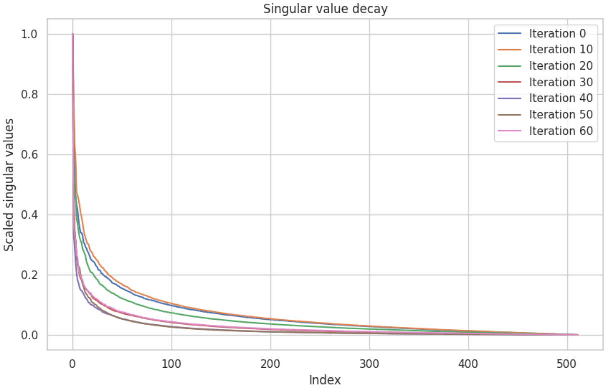


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

OBSERVING LOW RANK

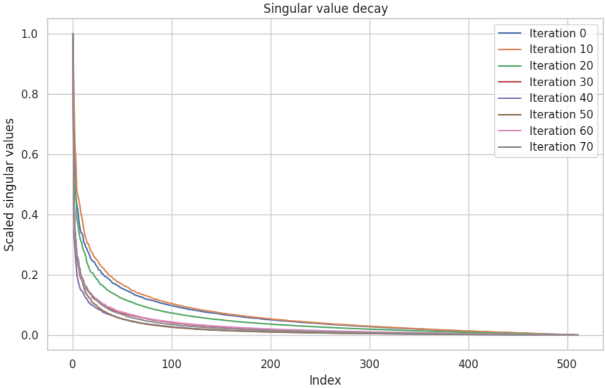


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

OBSERVING LOW RANK

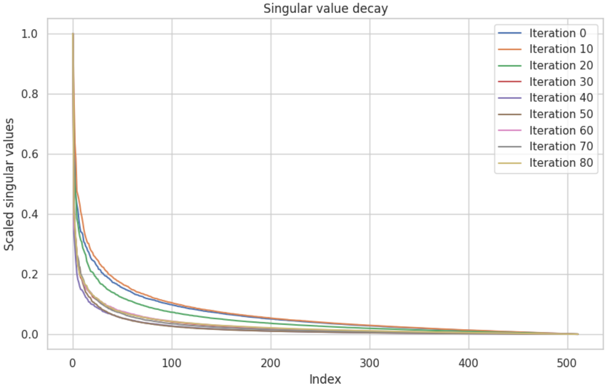


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

OBSERVING LOW RANK

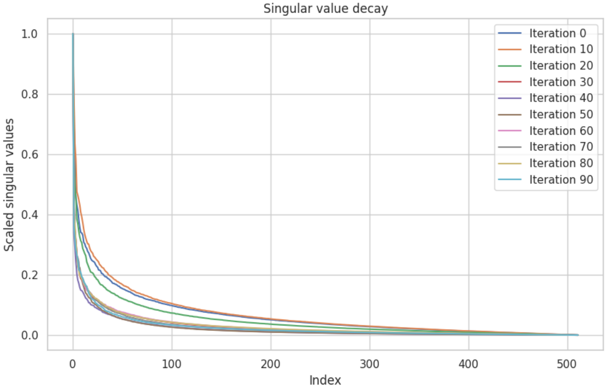


Figure: Singular value decay of gradient for specific layer in pre-training 60M Llama model after various iterations.

LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

- The weight updates of each layer are accumulated in two low-rank matrices.
- Mulitple LoRA adapters possible for multiple fine-tuned models from one base model.
- *r* is chosen a priori (as a hyperparameter).
- Not suited for pre-training.

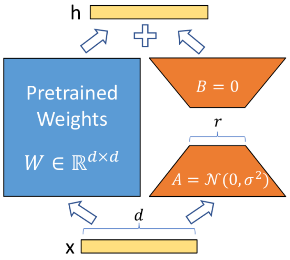


Figure 1: Our reparametrization. We only train A and B.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. "LoRA: Low-Rank Adaptation of Large Language Models", 2021.

GALORE: MEMORY-EFFICIENT LLM TRAINING BY GRADIENT LOW-RANK PROJECTION

Algorithm 2: Adam with GaLore

Input: A layer weight matrix $W \in \mathbb{R}^{m \times n}$ with $m \leq n$. Step size η, scale factor α, decay rates β_1, β_2, rank r, subspace change frequency T.

Initialize first-order moment $M_0 \in \mathbb{R}^{n \times r} \leftarrow 0$
Initialize second-order moment $V_0 \in \mathbb{R}^{n \times r} \leftarrow 0$
Initialize step $t \leftarrow 0$
repeat
  $G_t \in \mathbb{R}^{m \times n} \leftarrow -\nabla_W \varphi_t(W_t)$
  if $t \bmod T = 0$ **then**
    $U, S, V \leftarrow \text{SVD}(G_t)$
    $P_t \leftarrow U[:, :r]$ {Initialize left projector as $m \leq n$}
  else
    $P_t \leftarrow P_{t-1}$ {Reuse the previous projector}
  end if
  $R_t \leftarrow P_t^\top G_t$ {Project gradient into compact space}

  UPDATE(R_t) **by Adam**
    $M_t \leftarrow \beta_1 \cdot M_{t-1} + (1 - \beta_1) \cdot R_t$
    $V_t \leftarrow \beta_2 \cdot V_{t-1} + (1 - \beta_2) \cdot R_t^2$
    $M_t \leftarrow M_t / (1 - \beta_1^t)$
    $V_t \leftarrow V_t / (1 - \beta_2^t)$
    $N_t \leftarrow M_t / (\sqrt{V_t} + \epsilon)$

  $\tilde{G}_t \leftarrow \alpha \cdot P N_t$ {Project back to original space}
  $W_t \leftarrow W_{t-1} + \eta \cdot \tilde{G}_t$
  $t \leftarrow t + 1$
until convergence criteria met
return W_t

J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, Y. Tian. "GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection", 2024.

- Compute projection subspace every couple of iterations
- Compute full-rank gradient, then project it
- Update optimizer states (Momentum, Variance) with projected gradient.

→ $M_t, V_t \in \mathbb{R}^{m \times \ell}, \ell \ll n$

- Lower memory footprint than LoRA.
- Better suited for pre-training.

GALORE: MEMORY-EFFICIENT LLM TRAINING BY GRADIENT LOW-RANK PROJECTION

Algorithm 2: Adam with GaLore

Input: A layer weight matrix $W \in \mathbb{R}^{m \times n}$ with $m \leq n$. Step size η, scale factor α, decay rates β_1, β_2, rank r, subspace change frequency T.

Initialize first-order moment $M_0 \in \mathbb{R}^{n \times r} \leftarrow 0$

Initialize second-order moment $V_0 \in \mathbb{R}^{n \times r} \leftarrow 0$

Initialize step $t \leftarrow 0$

repeat

- $G_t \in \mathbb{R}^{m \times n} \leftarrow -\nabla_W \varphi_t(W_t)$
- **if** $t \bmod T = 0$ **then**
  - $U, S, V \leftarrow \text{SVD}(G_t)$
  - $P_t \leftarrow U[:, :r]$ {Initialize left projector as $m \leq n$}
- **else**
  - $P_t \leftarrow P_{t-1}$ {Reuse the previous projector}
- **end if**
- $R_t \leftarrow P_t^\top G_t$ {Project gradient into compact space}

- UPDATE(R_t) **by Adam**
  - $M_t \leftarrow \beta_1 \cdot M_{t-1} + (1 - \beta_1) \cdot R_t$
  - $V_t \leftarrow \beta_2 \cdot V_{t-1} + (1 - \beta_2) \cdot R_t^2$
  - $M_t \leftarrow M_t / (1 - \beta_1^t)$
  - $V_t \leftarrow V_t / (1 - \beta_2^t)$
  - $N_t \leftarrow M_t / (\sqrt{V_t} + \epsilon)$

- $\tilde{G}_t \leftarrow \alpha \cdot P N_t$ {Project back to original space}
- $W_t \leftarrow W_{t-1} + \eta \cdot \tilde{G}_t$
- $t \leftarrow t + 1$

until convergence criteria met

return W_t

J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, Y. Tian. “GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection”, 2024.

Computing the whole SVD is horribly inefficient, when all you want is an approximate basis of range(G_t).

THE RANDOMIZED RANGE FINDER

The right tool for the job

ALGORITHM 4.1: RANDOMIZED RANGE FINDER

Given an $m \times n$ matrix $\boldsymbol{A}$, and an integer ℓ, this scheme computes an $m \times \ell$ orthonormal matrix $\boldsymbol{Q}$ whose range approximates the range of $\boldsymbol{A}$.

1. Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$.
2. Form the $m \times \ell$ matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$.
3. Construct an $m \times \ell$ matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$, e.g., using the QR factorization $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$.

N. Halko, P.-G. Martinsson, J. A. Tropp. “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions”, 2010.

For an oversampling parameter $p \in \mathbb{N}$, $0 \leq p \leq r$, we have

$$\|A - QQ^T A\|_2 \leq \left(1 + 11\sqrt{r} \cdot \sqrt{\min\{m, n\}}\right) \sigma_{r-p+1} \qquad (1)$$

with a probability of at least $1 - 6 \cdot p^{-p}$ under mild assumptions on p.

THE RANDOMIZED RANGE FINDER

The right tool for the job

ALGORITHM 4.1: RANDOMIZED RANGE FINDER

Given an $m \times n$ matrix $\boldsymbol{A}$, and an integer ℓ, this scheme computes an $m \times \ell$ orthonormal matrix $\boldsymbol{Q}$ whose range approximates the range of $\boldsymbol{A}$.

1. Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$.
2. Form the $m \times \ell$ matrix $\boldsymbol{Y} = \boldsymbol{A\Omega}$.
3. Construct an $m \times \ell$ matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$, e.g., using the QR factorization $\boldsymbol{Y} = \boldsymbol{QR}$.

N. Halko, P.-G. Martinsson, J. A. Tropp. “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions”, 2010.

Plug this in and get a speedup of 10*x* and more compared to full SVD composition, with similar loss curve.

THE ADAPTIVE RANDOMIZED RANGE FINDER

- In later iterations, lower rank suffices for same approximation quality.
- Idea: Fix tolerance for subspace approximation instead of rank and use adaptive randomized rangefinder.

THE ADAPTIVE RANDOMIZED RANGE FINDER

- In later iterations, lower rank suffices for same approximation quality.
- Idea: Fix tolerance for subspace approximation instead of rank and use adaptive randomized rangefinder.
- Variant of classical Gram-Schmidt orthogonalization.

Algorithm 4.2: Adaptive Randomized Range Finder

Given an $m \times n$ matrix $\boldsymbol{A}$, a tolerance ε, and an integer r (e.g. $r = 10$), the following scheme computes an orthonormal matrix $\boldsymbol{Q}$ such that (4.2) holds with probability at least $1 - \min\{m,n\}10^{-r}$.

Draw standard Gaussian vectors $\boldsymbol{\omega}^{(1)}, \dots, \boldsymbol{\omega}^{(r)}$ of length n.
For $i = 1, 2, \dots, r$, compute $\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{\omega}^{(i)}$.
$j = 0$.
$\boldsymbol{Q}^{(0)} = [\ ]$, the $m \times 0$ empty matrix.
while $\max\left\{\left\|\boldsymbol{y}^{(j+1)}\right\|, \left\|\boldsymbol{y}^{(j+2)}\right\|, \dots, \left\|\boldsymbol{y}^{(j+r)}\right\|\right\} > \varepsilon/(10\sqrt{2/\pi})$,
    $j = j + 1$.
    Overwrite $\boldsymbol{y}^{(j)}$ by $\left(\mathbf{I} - \boldsymbol{Q}^{(j-1)}(\boldsymbol{Q}^{(j-1)})^*\right)\boldsymbol{y}^{(j)}$.
    $\boldsymbol{q}^{(j)} = \boldsymbol{y}^{(j)} / \left\|\boldsymbol{y}^{(j)}\right\|$.
    $\boldsymbol{Q}^{(j)} = [\boldsymbol{Q}^{(j-1)}\ \boldsymbol{q}^{(j)}]$.
    Draw a standard Gaussian vector $\boldsymbol{\omega}^{(j+r)}$ of length n.
    $\boldsymbol{y}^{(j+r)} = \left(\mathbf{I} - \boldsymbol{Q}^{(j)}(\boldsymbol{Q}^{(j)})^*\right)\boldsymbol{A}\boldsymbol{\omega}^{(j+r)}$.
    for $i = (j+1), (j+2), \dots, (j+r-1)$,
        Overwrite $\boldsymbol{y}^{(i)}$ by $\boldsymbol{y}^{(i)} - \boldsymbol{q}^{(j)}\left\langle \boldsymbol{q}^{(j)}, \boldsymbol{y}^{(i)} \right\rangle$.
    end for
end while
$\boldsymbol{Q} = \boldsymbol{Q}^{(j)}$.

N. Halko, P.-G. Martinsson, J. A. Tropp. “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions”, 2010.

A BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

- A blocked algorithm is required to effiently exploit the memory hierarchy of modern hardware, including GPUs.

$$A \begin{bmatrix} | & | & & | \\ \Omega_1 & \Omega_2 & \dots & \Omega_{n_b} \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ Q_1 & Q_2 & \cdots & Q_{n_b} \\ | & | & & | \end{bmatrix} \begin{bmatrix} - & B_1 & - \\ - & B_2 & - \\ & \vdots & \\ - & B_{n_b} & - \end{bmatrix}$$

1. Compute Q_1: $A\Omega_1 = Q_1 R_1, \quad B_1 = Q_1^T A$
2. Compute Q_2: $(I - Q_1 Q_1^T)A\Omega_2 = Q_2 R_2, \quad B_2 = Q_2^T A$
3. Compute Q_3: $(I - \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T)A\Omega_3 = Q_3 R_3, \quad R_2 = Q_3^T A$
4. ...

A BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

	function $[\mathbf{Q}, \mathbf{B}] = \texttt{randQB\_b}(\mathbf{A}, \varepsilon, b)$	
(1)	**for** $i = 1, 2, 3, \ldots$	
(2)	$\quad \mathbf{\Omega}_i = \texttt{randn}(n, b)$	
(3)	$\quad \mathbf{Q}_i = \texttt{orth}(\mathbf{A}\mathbf{\Omega}_i)$	$C_{\mathrm{mm}} mnb + C_{\mathrm{qr}} mb^2$
(3')	$\quad \mathbf{Q}_i = \texttt{orth}(\mathbf{Q}_i - \sum_{j=1}^{i-1} \mathbf{Q}_j \mathbf{Q}_j^* \mathbf{Q}_i)$	$2(i-1) C_{\mathrm{mm}} mb^2 + C_{\mathrm{qr}} mb^2$
(4)	$\quad \mathbf{B}_i = \mathbf{Q}_i^* \mathbf{A}$	$C_{\mathrm{mm}} mnb$
(5)	$\quad \mathbf{A} = \mathbf{A} - \mathbf{Q}_i \mathbf{B}_i$	$C_{\mathrm{mm}} mnb$
(6)	$\quad$ **if** $\lVert \mathbf{A} \rVert < \varepsilon$ **then stop**	
(7)	**end for**	
(8)	Set $\mathbf{Q} = [\mathbf{Q}_1 \ \cdots \ \mathbf{Q}_i]$ and $\mathbf{B} = [\mathbf{B}_1^* \ \cdots \ \mathbf{B}_i^*]^*$.	

P.-G. Martinsson, S. Voronin. “A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices”, 2015.

- Orthogonalizations are accumulated in *A*, which approaches zero and becomes a non-probabilistic error indicator.

$\rightarrow$ not ideal in memory-constrained environment.

- Reorthogonalization (line 3’) may be necessary in floating point arithmetic to ensure orthogonality of *Q*.

A HOUSEHOLDER BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

- Idea: Adaptively compute Householder-QR decomposition of $A \begin{bmatrix} \Omega_0 & \dots & \Omega_k \end{bmatrix}$ in factored form (`geqrt3`-style).

A HOUSEHOLDER BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

- Idea: Adaptively compute Householder-QR decomposition of $A \begin{bmatrix} \Omega_0 & \dots & \Omega_k \end{bmatrix}$ in factored form (`geqrt3`-style).
- V (lower triangular): contains Householder vectors representing Q, s.t. $QR = A \begin{bmatrix} \Omega_1 & \dots & \Omega_k \end{bmatrix}$

A HOUSEHOLDER BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

- Idea: Adaptively compute Householder-QR decomposition of $A \begin{bmatrix} \Omega_0 & \dots & \Omega_k \end{bmatrix}$ in factored form (`geqrt3`-style).
- V (lower triangular): contains Householder vectors representing Q, s.t. $QR = A \begin{bmatrix} \Omega_1 & \dots & \Omega_k \end{bmatrix}$
- A: Used to store B.

A HOUSEHOLDER BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

- Idea: Adaptively compute Householder-QR decomposition of $A\begin{bmatrix}\Omega_0 & \dots & \Omega_k\end{bmatrix}$ in factored form (`geqrt3`-style).
- V (lower triangular): contains Householder vectors representing Q, s.t. $QR = A\begin{bmatrix}\Omega_1 & \dots & \Omega_k\end{bmatrix}$
- A: Used to store B.
- T: Contains triangular blocks of storage-efficent QR decomposition of block reflectors

$$V = \begin{bmatrix} | & & \\ V_1 & & \\ | & & \end{bmatrix}, \; B = \begin{bmatrix} - & B_1 & - \\ & & \\ & & \end{bmatrix},$$

$$T = \begin{bmatrix} T_1 & & \end{bmatrix}$$

A HOUSEHOLDER BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

- Idea: Adaptively compute Householder-QR decomposition of $A\begin{bmatrix}\Omega_0 & \dots & \Omega_k\end{bmatrix}$ in factored form (`geqrt3`-style).
- V (lower triangular): contains Householder vectors representing Q, s.t. $QR = A\begin{bmatrix}\Omega_1 & \dots & \Omega_k\end{bmatrix}$
- A: Used to store B.
- T: Contains triangular blocks of storage-efficent QR decomposition of block reflectors

$$V = \begin{bmatrix} | & | & & \\ V_1 & V_5 & & \\ | & | & & \end{bmatrix}, \; B = \begin{bmatrix} - & B_1 & - \\ - & B_2 & - \\ & & \\ & & \end{bmatrix},$$

$$T = \begin{bmatrix} T_1 & T_2 & & \end{bmatrix}$$

A HOUSEHOLDER BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

- Idea: Adaptively compute Householder-QR decomposition of $A\begin{bmatrix}\Omega_0 & \dots & \Omega_k\end{bmatrix}$ in factored form (`geqrt3`-style).
- V (lower triangular): contains Householder vectors representing Q, s.t. $QR = A\begin{bmatrix}\Omega_1 & \dots & \Omega_k\end{bmatrix}$
- A: Used to store B.
- T: Contains triangular blocks of storage-efficent QR decomposition of block reflectors

$$V = \begin{bmatrix} | & | & & \\ V_1 & V_5 & \cdots & \\ | & | & & \end{bmatrix}, \; B = \begin{bmatrix} - & B_1 & - \\ - & B_2 & - \\ & \vdots & \\ & & \end{bmatrix},$$

$$T = \begin{bmatrix} T_1 & T_2 & \cdots & \end{bmatrix}$$

JÜLICH Forschungszentrum

A HOUSEHOLDER BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

- Idea: Adaptively compute Householder-QR decomposition of $A \begin{bmatrix} \Omega_0 & \dots & \Omega_k \end{bmatrix}$ in factored form (`geqrt3`-style).
- V (lower triangular): contains Householder vectors representing Q, s.t. $QR = A \begin{bmatrix} \Omega_1 & \dots & \Omega_k \end{bmatrix}$
- A: Used to store B.
- T: Contains triangular blocks of storage-efficent QR decomposition of block reflectors

$$V = \begin{bmatrix} | & | & & | \\ V_1 & V_5 & \cdots & V_k \\ | & | & & | \end{bmatrix}, \; B = \begin{bmatrix} - & B_1 & - \\ - & B_2 & - \\ & \vdots & \\ - & B_k & - \end{bmatrix},$$

$$T = \begin{bmatrix} T_1 & T_2 & \cdots & T_k \end{bmatrix}$$

A HOUSEHOLDER BLOCKED ADAPTIVE RANDOMIZED RANGE FINDER

- Stopping criterion: Use scalar error based on Frobenius norm from
  W. Yu, Y. Gu, Y. Li. “Efficient Randomized Algorithms for the Fixed-Precision Low-Rank Matrix Approximation”, 2018.

Algorithm 1: Householder block adaptive randomized range finder

---

Require: A matrix $A \in \mathbb{R}^{m \times n}$, a tolerance σ, and a block size b.

$E \leftarrow \|A\|_F$
$B \leftarrow A$
$i \leftarrow 1$
while $E > \sigma$ **do**
  Fill $\Omega \in \mathbb{R}^{n \times b}$ with values from a standard Gaussian distribution.
  $V_{i:,i}, T_i \leftarrow \mathrm{qr}(A_{i:,:}\Omega)$ {Storage-efficient QR decomposition, `geqrt`}
  $B_{i:,:} \leftarrow (I - V_i T_i V_i^T) B_{i:,:}$
  $E \leftarrow E - \|B_{i,:}\|_F$
  $i \leftarrow i + 1$
end while
$r = i - 1$
$V \leftarrow V_{:,0:r}$
$B \leftarrow B_{0:r,:}$

Ensure: $V \in \mathbb{R}^{m \times rb}$ Householder vectors, $B \in \mathbb{R}^{rb \times n}$,
  $T_0, \ldots, T_i \in \mathbb{R}^{b \times b}$ such that $A - QB < \sigma$, where
  $Q = \prod_{i=0}^{r} (I - V_i T_i V_i^T)$

HOUSEHOLDER VS. GRAM-SCHMIDT

Stability

- Modified Gram Schmidt:

$$Q^T Q = I + E_{MGS}, \quad \|E_{MGS}\|_2 \approx u\kappa_2(A)$$

- Householder QR:

$$Q^T Q = I + E_H, \quad \|E_H\|_2 \approx u$$

- We observed in our application, $\kappa_2(A)$ will become very large.

Operation count

- In factored form (Householder) and without reorthogonalization (Gram-Schmidt) both take around $2mn^2$ operations.

Practical issues

- GPU-based, optimized version for Householder-QR are available (MAGMA library) and can be adapted for the rangefinder algorithm.

OVERLAP COMMUNICATION, COMPUTATION AND RANDOM GENERATION

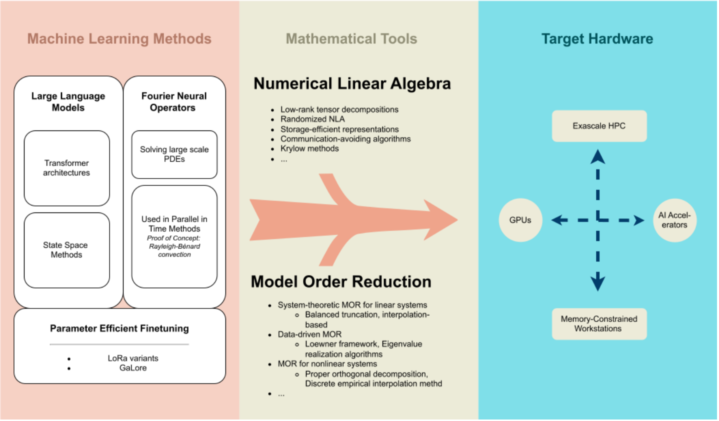
Queue 1	Queue 2	Queue 3
		Create Ω_1
	$V_1 \leftarrow A\Omega_1$	Create Ω_2
$V_1, T_1 \leftarrow qr(V_1)$	$V_2 \leftarrow A\Omega_2$	
$V_2 \leftarrow (I - V_1 T_1 V_1^T) V_2$	$A \leftarrow (I - V_1 T_1 V_1^T) A$	Create Ω_3
$V_2, T_2 \leftarrow qr(V_2)$	$V_3 \leftarrow A\Omega_3$	
$V_3 \leftarrow (I - V_2 T_2 V_2^T) V_3$	$A \leftarrow (I - V_2 T_2 V_2^T) A$	Create Ω_4
$V_3, T_3 \leftarrow qr(V_3)$	$V_4 \leftarrow A\Omega_4$	
$\vdots$	$\vdots$	$\vdots$

- More operations (explicit panel update) in favor of exposed parallelism.

FURTHER RESEARCH

- Experiments and results.
- Relative vs. absolute stopping criterion.
- How do stability results translate to randomized setting.
- Two-sided projections
- Mix Gram-Schmidt and QR
- Cholesky QR
- Other decompositions from Randomized Numerical Linear Algebra
- Extend to higher dimensional tensors.
- Formalize relationship between LoRA and Galore
- How to find a good rank for LoRA?

MORE INTERESTING RESEARCH OPPORTUNITIES



Thank you for your attention!