MEMORY-EFFICIENT GRADIENT DESCENT WITH RANDOMIZED EIGENVALUE SKETCHING

FINAL PROJECT

Johannes Losert
Department of Computer Science
Columbia University
New York, NY 10027
jal2340@columbia.edu

Mark Kirichev
Department of Computer Science
Columbia University
New York, NY 10027
mmk2243@columbia.edu

Alexia Popescu
IEOR Department
Columbia University
New York, NY 10027
ap4314@columbia.edu

October 4, 2024

ABSTRACT

This survey has two main parts. First we will show how to reduce the amount of memory consumed by gradient descent optimizer states for specific types of functions/neural networks. We will show that the rank of the gradient decays exponentially fast during the optimization of 'reversible' neural networks. Thereafter we will show how this property is leveraged to reduce memory consumption with the GaLore algorithm.

In the second part of the survey, we outline an algorithm for approximating the rank, eigenvalues, and singular value decomposition of the gradient matrix in GaLore. We contribute to an existing result by showing how a coarse eigenvalue sketch can be attained with a random sign matrix.

1 Introduction

We will start by presenting the main results from [1], which proposes a memory-efficient algorithm for the optimization of stochastic objective functions. [1] shows that the gradient of a specific class of *reversible* functions becomes low rank during optimization. Furthermore, it proves that convergence guarantees are retained for optimizing just a subset of the parameters. This fact is used to reduce the memory consumption of Adam [2] optimizer states in the improved GaLore algorithm.

We will leverage [3], which gives an efficient random algorithm for computing the numerical rank of a matrix. We show how to estimate the rank in order to remove the dependence on the rank as a hyperparameter in [1]. Furthermore, we propose that we don't have to compute the true singular value decomposition of the gradient, since the exact structure of the left projector matrix is never leveraged. With this in mind, we invoke a variant of randomized singular value decomposition [4] that is proposed in [3] to compute the projector matrix.

2 Problem Setting

Adaptive moment estimation (Adam) [2] is the most popular optimization algorithm for modern machine learning. For a stochastic objective function $f(\theta)$ with a gradient $g(\theta)$, Adam maintains estimates of $\hat{v} \approx \mathbb{E}[g(\theta)^2]$ and $\hat{m} \approx \mathbb{E}[g(\theta)]$ in order to perform gradient steps of the form: $s = \frac{\hat{m}}{\sqrt{\hat{v}}}$, where the division is element-wise. Thus, in order to use Adam, you must store $2\|\theta\|$ optimizer states in addition to $\|\theta\|$ 'activations' for the backpropagation algorithm. When $\|\theta\|$ is very large, however, it can become impossible to store all $3\|\theta\|$ parameters of the model in memory. However, if we know that $f(\theta)$ has a **reversible** structure, we can reduce the amount of memory needed by compressing the optimizer state. We seek such fine-grained (multiplicative) reductions in memory usage, because it can allow us to optimize these functions on much cheaper hardware.

3 Reversibility

Reversibility, first discussed in [1], is a framework with which we can analyze the convergence of a broad class of composable functions. A network (composition of functions) $\mathcal{N}(x)$ is reversible if there exists a $K(x;W)$ so that $y = K(x,W)$ and the backpropagated gradient satisfies $g_x = K^\top(x;W)g_y$. K is a linear map that depends on both the input x and the weight W of the network. Essentially, we are focusing on whether the forward propagation through the network can be completely reversed by appropriately defined backward operations, ensuring no loss of information.

The composition of reversible functions can be rewritten as a product of appropriately parameterized matrices. The subscript $\mathcal{N}_i(x)$ denotes the 'layer' in which the function resides, which can be understood as its order in the composition.

$$\mathcal{N}(x) = \mathcal{N}_L(\mathcal{N}_{L-1}(...\mathcal{N}_1(x))) = K_L(x)K_{L-1}(x)\dots K_1(x)x \tag{1}$$

In order to figure out how the output of the network changes with respect to the input, we must compute the Jacobian. Since $K(x;W)_L$ is linear, the Jacobian of $\mathcal{N}(x)$ is also equal to $K(x;W)_L$. Therefore, the Jacobian of the network with respect to layer L can be written as:

$$J_l = K_L(x)K_{L-1}(x)\dots K_{l+1}(x)x$$

.

3.1 What Networks are Reversible?

[1] shows that the functions that generally satisfy the *reversability* property need to have certain properties. Examples include ReLU, LeakyReLU, monomial functions, and the exponential function $\psi(x) := x^p$ where $p > 1$. However, at this point we would like to inspect **softmax**, due to its prevalence in modern machine learning architectures such as transformers:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

This causes some problems as the softmax function has a stochastic nature. By Property 1 in the Appendix A.2. of Galore, we note that if $\mathcal{N}_1, \mathcal{N}_2$ are reversible then $\mathcal{N}_1(\mathcal{N}_2)$ is reversible and $\alpha_1\mathcal{N}_1 + \alpha_2\mathcal{N}_2$ for **constant** α_i. Thus, we can try to argue that the denominator can be seen as a sum of exponentials and be represented as $\alpha_i := 1, \mathcal{N} := x^p$ form. This assumes that p is some constant which ultimately will disrupt the stochastic property. The same argument is valid for the numerator as it can also be seen as a sum of exponentials the following way:

$$e^{z_i} := 1 \cdot e^{z_i}$$

The division can be overcome by ... (some argument here will be added in the next couple of hours). It follows that the only problem arises when we try to represent the quantity e^{z_i} as a function of exponential raised to a *constant* power.

Challenges in Reversibility

1. **Non-injectivity**: The softmax function is not injective (one-to-one), meaning different input vectors $\mathbf{z}$ can produce the same output probabilities. This non-injectivity is a fundamental barrier to reversibility since a unique inverse function does not exist.
2. **Dependency on All Inputs**: The output of softmax for a particular class depends on all elements of the input vector $\mathbf{z}$, due to the normalization step (the denominator). This interdependence complicates any attempt to isolate the effect of a single input on the output, contrasting with functions like ReLU, which act independently on each input component.
3. **Stochastic Nature**: As highlighted, the softmax function's output is inherently probabilistic. This stochastic nature means that even slight variations in input can lead to disproportionately large changes in output, further complicating the derivation of a stable inverse function.

Mathematical Constraints and Properties: To explore the possibility of overcoming these challenges, consider the properties of exponential functions and their sums. The exponential function e^x is always positive and increases exponentially, which is a key aspect utilized by softmax to amplify differences among input values. However, this also means that the softmax function is highly sensitive to the largest values in $\mathbf{z}$, overshadowing smaller ones, which is another aspect that complicates its inversion.

Theoretical Considerations for Partial Reversibility: While a full analytical inversion of softmax may not be feasible due to the reasons stated above, it might be possible to approximate the inverse under constrained conditions or by using additional information about the distribution of inputs. For instance, if the range of input values $\mathbf{z}$ is tightly controlled or known a priori, one could potentially estimate the original logits from the softmax probabilities by solving a system of equations derived from the softmax function's definition. This approach, however, would require strong assumptions about the input data and might not be generally applicable.

3.2 Gradient Structure of Reversible Networks

It is showin in Theorem 3.2 of the GaLore paper that the gradient structure of reversible networks can be written in a special form. We will expand on this proof with our own annotations for enhanced clarity. We will derive the gradient of a reversible network for the loss function. $\phi = \frac{1}{2}\|y - \mathcal{N}(x)\|_2^2$. We will walk through a computation of the gradient $\frac{d\phi}{dW_l}$.

First we apply the chain rule for matrices, which removes the dependence on the square.

$$d\phi = (\mathbf{y} - \mathcal{N}(\mathbf{x}))^\top d\mathcal{N}(\mathbf{x}) \tag{2}$$

We now define $f_l = \mathcal{N}(x)_l \mathcal{N}(x)_{l-1} \cdots \mathcal{N}(x)_1$ as the output at layer l. Furthermore, we denote $dK(x;W)_l$ as dW_l for clarity. Now we can write $d\mathcal{N}(x)_l$ as $\sum_i^L J_i dW_i f_{i-1}$. Since we are only focusing on layer l we ignore the other terms in the summation.

$$= (\mathbf{y} - \mathcal{N}(\mathbf{x}))^\top K_L(\mathbf{x}) \dots K_{l+1}(\mathbf{x}) dW_l \mathbf{f}_{l-1} \tag{3}$$

$$= (\mathbf{y} - J_l W_l \mathbf{f}_{l-1})^\top J_l dW_l \mathbf{f}_{l-1} = \mathrm{tr}((\mathbf{y} - J_l W_l \mathbf{f}_{l-1})^\top J_l dW_l \mathbf{f}_{l-1}) = \mathrm{tr}((J_l dW_l \mathbf{f}_{l-1})^\top (\mathbf{y} - J_l W_l \mathbf{f}_{l-1})) \tag{4}$$

$$= \mathrm{tr}(dW_l^\top J_l^\top (\mathbf{y} - J_l W_l \mathbf{f}_{l-1}) \mathbf{f}_{l-1}^\top) \tag{5}$$

Finally, with the cyclic property of trace we have:

$$= \mathrm{tr}(dW_l^\top (J_l^\top (\mathbf{y} - J_l W_l \mathbf{f}_{l-1}) \mathbf{f}_{l-1}^\top)) \tag{6}$$

Finally we reason that the part which is not $dW_l^\top$ must be the gradient.

$$\frac{d\phi}{dW_l} = G_l = J_l^\top \mathbf{y} \mathbf{f}_{l-1}^\top - J_l^\top J_l W_l \mathbf{f}_{l-1} \mathbf{f}_{l-1}^\top \tag{7}$$

This gradient can be expressed as $G = A - BW_tC$, where $A = J_l^\top \mathbf{y} \mathbf{f}_{l-1}^\top$ is a constant matrix for a fixed timestep t, and $B = J_l^\top J_l, C = \mathbf{f}_{l-1} \mathbf{f}_{l-1}^\top$ are positive semidefinite. With this structure, we can prove that the rank of G decays exponentially in the next section.

3.3 Gradients of Reversible Networks Decay in Rank

We include beginning of this proof from [1] as it is all that is necessary for the intuition. The unrolling of the recursion is the most important part:

$$G_t = A - BW_tC = A - B(W_{t-1} + \eta G_{t-1})C = G_{t-1} - \eta B G_{t-1} C \tag{8}$$

Let $B = UD_BU^T$ and $C = VD_CV^T$ be the eigendecomposition of B and C (which exists because they are PSD). $D_B = \mathrm{diag}(\lambda_1, \dots, \lambda_m)$ and $D_C = \mathrm{diag}(v_1, \dots, v_n)$ are their eigenvalues sorted in ascending orders (i.e., $\lambda_1 \le \cdots \le \lambda_m$ and $v_1 \le \cdots \le v_n$). Define $H_t := U^T G_t V$. It is clear that $\mathrm{rank}(H_t) = \mathrm{rank}(G_t)$ and we have:

$$H_t = U^T G_t V = H_{t-1} - \eta D_B H_{t-1} D_C$$

Suppose $h_{t,ij}$ is the ij component of H_t, then from the equation above we have:

$$h_{t,ij} = h_{t-1,ij} - \eta \lambda_i v_j h_{t-1,ij} = (1 - \eta \lambda_i v_j) h_{t-1,ij} = (1 - \eta \lambda_i v_j)^t h_{0,ij}$$

Then for first few rows i and columns j that correspond to large eigenvalues, $h_{t,ij} \to 0$ quickly and $\mathrm{rank}(H_t)$ becomes small.

3.4 GaLore Algorithm

Below we have included the GaLore algorithm as it is specified in [1]. We draw attention to the hyperparameters T and r. Having T as a hyperparameters is almost a necessity, because otherwise have to compute the randomized SVD in every gradient step, which takes time $O(r^3)$. This becomes prohibitively expensive unless $r << m$. However, we emphasize that r must in general not be a hyperparameter, and that we can find an appropriate value for it *while* computing the singular value decomposition. Thus, we introduce the notion of the ϵ-rank of a matrix in the second part of this survey as the appropriate setting of r. Of course, if we are in a computational model, where we do not care about speed but must ensure that memory capacity is not exceeded, it makes sense to fix r to its highest possible value. However, in general the cost of computing the singular value decomposition is high enough for us to want r as small as possible in every gradient step. Namely, we face a cubic speed penalty and quadratic memory penalty for overestimating r, which we can mitigate if we construct the singular value decomposition in a smart way.

Algorithm 1: 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
  $G_t \in \mathbb{R}^{m \times n} \leftarrow -\nabla_W \phi_t(W_t)$
  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**
    $M_t \leftarrow \beta_1 \cdot M_{t-1} + (1 - \beta_1) \cdot R_t$
    $V_t \leftarrow \beta_2 \cdot V_{t-1} + (1 - \beta_2) \cdot R_t^2$
    $M_t \leftarrow M_t / (1 - \beta_1)$
    $V_t \leftarrow V_t / (1 - \beta_2)$
    $N_t \leftarrow M_t / (\sqrt{V_t} + \epsilon)$

  $\tilde{G}_t \leftarrow \alpha \cdot P N_t$ {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

4 Fast Randomized Numerical Rank Estimation

Given a hyperparameter r, the Galore algorithm computes the singular value decomposition of G in time $O(r^3)$ and takes r left singular vectors to form a projector matrix P. However, since r is a hyperparameter, it is easily possible for it to be too small or too large. Therefore, we can benefit greatly from an algorithm that can compute the rank r quickly in order to prevent superfluous computations. For this task, we apply results from [3]. This paper sketches the eigenvalues of a matrix A by picking a fully random X matrix, and a structured random Y matrix such that $\text{rank}(A) \approx \text{rank}(YAX)$. Instead of trying to estimate the 'true' rank of A, the paper instead defines the ϵ-rank or 'numerical rank' or as its quantity of interest. Overall the runtime for obtaining the singular value estimates is shown to be $O(mnlogn + r^3)$ for a $m \times n$ matrix of numerical rank r.

Definition 1. Let $A \in \mathbb{R}^{m\times n}$. The ϵ-rank of A, denoted by $\text{rank}_\epsilon(A)$, is the integer i such that $\sigma_i(A) \geq \epsilon\|A\|_2 \geq \sigma_{i+1}(A)$, where $\|A\|_2 = \sup_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}$.

4.1 Sketching Eigenvalues with Random Gaussian Matrices

The main theoretical contribution of this paper is the fact that $\frac{\sigma_i(AX)}{\sigma_i(A)} = O(1)$ for the leading i eigenvalues. This is an immediate consequence of Theorem 1.

Theorem 1:

$$\sigma_{\min}(\tilde{G}_i) \leq \frac{\sigma_i(AG)}{\sigma_i(A)} \leq \sqrt{\sigma_{\max}(\tilde{G}_{r-i+1})^2 + \left(\frac{\sigma_{r+1}(A)\sigma_{\max}(G_2)}{\sigma_i(A)}\right)^2},$$

where $\tilde{G}_i$ is an $i \times r$ matrix consisting of the first i rows of G_1, and $\tilde{G}_{r-i+1}$ is the matrix of the last $r - i + 1$ rows of G_1. Furthermore, if G is a Gaussian matrix, then for each $i \in \{1, \ldots, r\}$, the random matrices $\tilde{G}_i$, $\tilde{G}_{r-i+1}$, and G_2 are independent Gaussian in $\mathbb{F}$.

[3] splits the singular value decomposition of A into a part containing the leading singular values and a remainder.

$$AG = U_1\Sigma_1(V_1^\top G) + U_2\Sigma_2(V_2^\top G) \tag{9}$$

After making this distinction, [3] invokes the rotational invariance of the Gaussian distribution. We note that another distribution which is rotationally invariant is the uniform distribution of points on the surface of an n-dimensional unit sphere. However, in practice this distribution is generated by normalizing an entrywise-Gaussian vector to unit length.

$$AG = U_1\Sigma_1 G_1 + U_2\Sigma_2 G_2 \tag{10}$$

After establishing the above inequality, [3] uses probabilistic bounds on the extremal eigenvalues of random Gaussian matrices. Since the random matrix is only generated once, the failure probability of the algorithm is just the probability that the random Gaussian matrix does not fall into the bounds.

$$\sqrt{m} - \sqrt{n} \leq \mathbb{E}[\sigma_{\min}(G)] \leq \mathbb{E}[\sigma_{\max}(G)] \leq \sqrt{m} + \sqrt{n}. \tag{11}$$

$$\mathbb{P}\{\sigma_{\min}(G) \leq \sqrt{m} - \sqrt{n} - t\} \leq e^{-t^2/2}, \quad \mathbb{P}\{\sigma_{\max}(G) \geq \sqrt{m} + \sqrt{n} + t\} \leq e^{-t^2/2}. \tag{12}$$

We note that (11) holds in general for any random matrix with zero mean and unit variance entries.

4.2 Sketching Eigenvalues with Random Sign Matrices

Our group had the idea to construct the sketch matrix X with Rademacher distributions (uniformly over $\{-1, 1\}$). Such a matrix is also known as a 'random sign matrix', and it has the special property that the matrix multiplication AX is only additions and subtractions. We were inspired by results such as [5], which shows how Johnson-Lindenstrauss guarantees can be achieved with random sign matrices. Random sign matrices occupy much less memory and are inexpensive to generate. Unfortunately, the proofs for algorithms with random sign matrices are more involved, because in general such discrete distributions such as these are not rotationally invariant. Namely, the entries of the matrix after rotation might have non-discrete entries.

A useful fact about the Rademacher distribution is that it is subgaussian. A random variable is subgaussian if it has tail behavior and concentration properties that are similar to or decay at least as fast as those of a Gaussian random variable, despite possibly not having a Gaussian distribution. The properties of random matrices generated by subgaussian distributions have been studied thoroughly in works such as [6]. Below is a useful result that we will use later:

Lemma 1:

For any $N \times n$ random matrix A with independent subgaussian entries:

$$\mathbb{P}\left(\sigma_{\max}(A) \geq \sqrt{N} + \sqrt{n} + \tau\sqrt{N}\right) \leq C \exp\left(-cn\tau^{3/2}\right) \quad \text{for} \quad \tau \geq 0$$

The foundation of our proof will be the result from [3]. In particular, this framework will let us prove the efficacy of the random sign matrix without thinking about rotational invariance.

Theorem 2:

Let $V_1 \in \mathbb{F}^{n\times r}$ be the leading r right singular vectors of $A \in \mathbb{F}^{m\times n}$, and suppose $X \in \mathbb{F}^{n\times r}$ is a subspace embedding for the subspace V_1 satisfying $\sigma_i(V_1^\top X) \in [1-\varepsilon, 1+\varepsilon]$ for some $\varepsilon < 1$. Then, for $i = 1, \ldots, r'$,

$$1 - \varepsilon \leq \frac{\sigma_i(AX)}{\sigma_i(A)} \leq \sqrt{(1+\varepsilon)^2 + \left(\frac{\sigma_{r+1}(A)\|X\|_2}{\sigma_i(A)}\right)^2}$$

We can use results from approximate matrix multiplication in [7] to establish properties of random sign matrices. In this line of work, they are interest in randomized approximation algorithms to to the rank k factorization of a matrix. Indirectly they provide some immensely useful results:

Theorem 3:

Let $0 < \varepsilon < \frac{1}{2}$ and let $A \in \mathbb{R}^{n\times m}$, $B \in \mathbb{R}^{n\times p}$ both having rank and stable rank at most r and $\tilde{r}$, respectively. The following holds: Let R be a $t \times n$ random sign matrix rescaled by $1/\sqrt{t}$. Denote by $\tilde{A} = RA$ and $\tilde{B} = RB$. If $t = \Omega(r/\varepsilon^2)$, then

$$\mathbb{P}(\forall x \in \mathbb{R}^m, \forall y \in \mathbb{R}^p, |x^\top(\tilde{A}^\top\tilde{B} - A^\top B)y| \leq \varepsilon\|A\|_2\|B\|_2) \geq 1 - e^{-\Omega(r)}.$$

Lemma 2:

Let $0 < \epsilon \leq 1$. Assume $A, \hat{A}$ are $n \times n$ symmetric positive semi-definite matrices, such that the following inequality holds

$$(1-\epsilon)x^\top Ax \leq x^\top \hat{A}x \leq (1+\epsilon)x^\top Ax, \quad \forall x \in \mathbb{R}^n.$$

Then, for $i = 1, \ldots, n$ the eigenvalues of A and $\hat{A}$ are the same up-to an error factor ϵ, i.e.,

$$(1-\epsilon)\sigma_i(A) \leq \sigma_i(\hat{A}) \leq (1+\epsilon)\sigma_i(A).$$

Lemma 3 comes directly from [7], and it is the backbone of our final theorem.

Lemma 3:

Set $\tilde{A} = \frac{1}{\sqrt{t}}RA$ where R is a $t \times n$ random sign matrix, and pick $t = \Omega(r/\epsilon^2)$. By applying Theorem 3 on A, we have with high probability that

$$\forall x \in \mathbb{R}^n, \quad (1-\varepsilon)x^\top A^\top Ax \leq x^\top \tilde{A}^\top \tilde{A}x \leq (1+\varepsilon)x^\top A^\top Ax. \tag{13}$$

Finally, we can prove that the random sign matrix preserves the singular values of the right singular matrix.

Theorem 4 - Original Contribution:

Let $V_1 \in \mathbb{F}^{n\times r}$ be the leading r right singular vectors of $A \in \mathbb{F}^{m\times n}$. Set $\tilde{A} = \frac{1}{\sqrt{t}}RA$ where R is a $t \times n$ random sign matrix, and pick $t = \Omega(r/\epsilon^2)$. Then with high probability $|\sigma_i(V_1^\top R)| \in [1-\epsilon, 1+\epsilon]$.

Proof. Combining Corollary 1 and Lemma 1, we have with high probability that:

$$(1-\epsilon)\sigma_i(A^\top A) \leq \sigma_i(\tilde{A}^\top\tilde{A}) \leq (1+\epsilon)\sigma_i(A^\top A).$$

Since the singular values of $A^\top A$ are the squared singular values of A, we have:

$$(1-\epsilon)\left|\sigma_i(A)\right| \leq \left|\sigma_i(RA)\right| \leq (1+\epsilon)\left|\sigma_i(A)\right|.$$

By the fact that the eigenvalues of A are the eigenvalues of $A^\top$, we have

$$(1-\epsilon)\left|\sigma_i(A^\top)\right| \leq \left|\sigma_i(A^\top R^\top)\right|\right| \leq (1+\epsilon)\left|\sigma_i(A^\top)\right|.$$

We note that $R^\top$ is simply a random sign matrix of different dimension:

$$(1-\epsilon)\left|\sigma_i(A)\right| \leq \left|\sigma_i(AR)\right| \leq (1+\epsilon)\left|\sigma_i(A)\right|.$$

Setting $A = V_1$, and since $\forall i, \sigma_i(V_1) = 1$, we end with the fact that:

$$(1-\epsilon) \leq \left|\sigma_i(V_1 R)\right| \leq (1+\epsilon).$$

□

Now, we bring it all together to prove Theorem 5, which bounds the precision of our eigenvalue sketch with the random sign matrix.

Theorem 5 - Original Contribution:

Suppose R is a $n \times \Omega(r/\varepsilon^2)$ random sign matrix rescaled by $\frac{1}{\sqrt{n}}$. Then, with high probability, for any $m \times n$ matrix A and for $i = 1, \ldots, r'$,

$$1-\varepsilon \leq \left|\frac{\sigma_i(AR)}{\sigma_i(A)}\right| \leq \sqrt{(1+\varepsilon)^2 + \left(\frac{\sigma_{r+1}(A)\|R\|_2}{\sigma_i(A)}\right)^2}$$

Applying the probabilistic bound in (13), we can upper bound $\|R\|_2$. We note that this bound doesn't immediately happen with high probability. However, as the size of A increases, the failure probability of our randomized eigenvalue sketch decays exponentially.

$$1-\varepsilon \leq \left|\frac{\sigma_i(AR)}{\sigma_i(A)}\right| \leq \sqrt{(1+\varepsilon)^2 + \left(\frac{\sigma_{r+1}(A)(\sqrt{m}(i+1)+\sqrt{n})}{\sigma_i(A)}\right)^2}$$

Proof. Follows immediately by combining Theorem 2, Theorem 4, and Lemma 1. □

The main sacrifice we observe from using random sign matrix embeddings, is that we lose information about the signs of the individual singular values. Further work would be needed to determine bounds on the probability of a sign flip in the singular values, because we would guess that it is very low. Thus, we have shown that random sign matrix embeddings preserve the magnitude of the singular values in the embedded matrix and can be computed without slow floating point multiplication!

4.3 Integration of Randomized Rangefinder for SVD Computation

The **Singular Value Decomposition (SVD)** is a fundamental tool in numerical linear algebra and machine learning, particularly for matrix approximation and dimensionality reduction. In the context of the GaLore algorithm, efficient computation of the SVD is crucial for handling large matrices G that arise during gradient updates. To address the computational and storage challenges associated with direct SVD computations, we employ the Randomized Rangefinder technique, which effectively reduces the dimensionality of the problem while preserving the essential spectral properties of the matrix.

4.3.1 Mathematical Preliminaries

The Randomized Rangefinder method approximates the range (column space) of a matrix $G \in \mathbb{R}^{m\times n}$ using a smaller matrix that captures the dominant actions of G. The process involves two main steps: random projection and orthogonalization.

Random Projection: We start by choosing a random projection matrix $\Omega \in \mathbb{R}^{n\times k}$, where $k \ll n$ is a small integer that represents the target number of dimensions. This matrix Ω is typically populated with Gaussian random variables due to their favorable properties, such as isotropy and density [8]. The projection is computed as:

$$Y = G\Omega$$

Here, Y represents a $m \times k$ matrix that is expected to span the range of the most significant singular vectors of G if k is chosen appropriately relative to the spectral properties of G.

Orthogonalization: To ensure that the columns of Y form an orthonormal basis without redundancy, we apply the QR decomposition:

$$Y = QR$$

where $Q \in \mathbb{R}^{m\times k}$ has orthonormal columns and $R \in \mathbb{R}^{k\times k}$ is an upper triangular matrix. The matrix Q now serves as an orthonormal basis for the column space of Y and, by extension, an approximation to the column space of G.

4.3.2 Approximating the SVD

With Q computed, we approximate G as:

$$\tilde{G} = Q(Q^T G) = QQ^T G$$

where $Q^T G$ is a smaller $k \times n$ matrix that we need to decompose further. The computational advantage here lies in the reduced size of the matrix for which the SVD is directly computed, namely $Q^T G$ instead of G.

To proceed with the SVD of $Q^T G$, we compute:

$$Q^T G = U\Sigma V^T$$

where $U \in \mathbb{R}^{k\times k}$ contains the left singular vectors, $\Sigma \in \mathbb{R}^{k\times k}$ is a diagonal matrix with the singular values, and $V \in \mathbb{R}^{n\times k}$ contains the right singular vectors of the projected matrix $Q^T G$. Thus, the approximate SVD of G becomes:

$$G \approx QU\Sigma V^T$$

This factorization efficiently captures the dominant singular values and vectors of G with reduced computational overhead.

4.3.3 Choice of Projection Dimension k

The effectiveness of the Randomized Rangefinder method hinges on the appropriate choice of k, the number of projection dimensions. It is crucial that k is chosen to exceed the effective numerical rank of G, denoted as r, to ensure that the dominant subspace is captured. The choice of k is typically $k = r + p$, where p is a small oversampling parameter ($p \approx 5 - 10$) to compensate for the probabilistic nature of the random projection and to improve the accuracy of the capture of the subspace [9].

Given the properties of reversible networks discussed in earlier sections, where the gradient matrices tend to have low-rank structures especially after several iterations of gradient updates, adaptive strategies can be employed to dynamically adjust k based on real-time assessments of matrix rank decay [10]. This can lead to further computational savings and optimized memory usage.

4.3.4 Algorithmic Complexity

The computational complexity of the Randomized Rangefinder approach for SVD computation is influenced primarily by the matrix-matrix multiplication and the QR decomposition steps involved. Specifically:

1. **Initial Projection:** Computing the projection $Y = G\Omega$ where Ω is a $n \times k$ Gaussian random matrix involves $O(mnk)$ operations. This step is essential for reducing the dimensionality of G from n to k, which is significantly smaller ($k \ll n$).
2. **QR Decomposition:** The QR decomposition of Y to obtain an orthonormal basis Q requires $O(mk^2)$ operations. This step ensures that the columns of Q are orthonormal, providing a stable basis for the subsequent approximation of G.
3. **Decomposition of Projected Matrix:** The SVD of the smaller matrix $Q^T G$ (which is $k \times n$) needs $O(k^2 n)$ operations. This is where the actual SVD is performed on a much-reduced scale compared to G.

Adding these together, the total computational complexity of the Randomized Rangefinder method for SVD can be estimated as $O(mnk + mk^2 + k^2 n)$. This combined complexity is indeed dominated by the initial projection step, and for k chosen much smaller than n, it is significantly less than the $O(mn \min(m, n))$ required for a direct SVD computation on the full matrix G.

4.3.5 Optimizing QR Decomposition

Recent advancements in numerical linear algebra have led to significant improvements in the efficiency of QR decomposition algorithms, particularly through the use of randomized algorithms and parallel computing techniques. These advancements can be leveratively integrated into the Randomized Rangefinder method to further reduce the computational overhead.

Advancements in QR Algorithms: Modern approaches to QR decomposition leverage randomized techniques that reduce the computational complexity by focusing on approximations that preserve the essential characteristics of the original matrix [11]. For example, by employing a block low-rank format within the QR decomposition process, the number of flops required can be significantly reduced, particularly when the matrix has inherent low-rank structure, which is often the case in matrices processed in machine learning applications.

Parallelization: In addition to algorithmic improvements, the utilization of parallel processing frameworks can substantially accelerate the QR decomposition step. Implementations using GPUs or distributed computing frameworks can exploit the inherent parallelism in the QR algorithm, particularly in the formation of Householder vectors and the subsequent matrix updates [12].

Impact on Total Runtime: By integrating these advanced QR decomposition techniques, the $O(mk^2)$ complexity of the orthogonalization step in the Randomized Rangefinder method can be significantly reduced. This reduction is particularly impactful because it directly decreases the second largest term in the overall computational complexity, following the initial projection step. For example, if the enhanced QR decomposition algorithms reduce the complexity of this step by a factor of γ, the new complexity for the QR step becomes $O(\gamma mk^2)$ with $\gamma < 1$, effectively lowering the total runtime of the Randomized Rangefinder method.

4.3.6 Randomized Rangefinder Overall Benefits

The integration of the Randomized Rangefinder method within the GaLore algorithm framework provides a computationally efficient and scalable way to approximate the SVD of large matrices encountered in advanced algorithmic contexts. By significantly reducing both computational time and memory requirements, this method facilitates more practical implementations of SVD computations in large-scale machine learning and data analysis tasks, especially when dealing with matrices where the dimensionality n is large.

5 Conclusion

This research has made significant contributions to the field of gradient descent optimizations for reversible functions, which are particularly relevant in the context of large-scale machine learning models and data-intensive applications. Our investigation centered on the GaLore **algorithm**, which capitalizes on the low-rank nature of gradients in such functions to dramatically reduce memory consumption.

The core of our contributions lies in the innovative application of randomized algorithms for matrix decompositions, which we tailored to enhance the efficiency and scalability of the GaLore method. We introduced a novel approach utilizing random sign matrices to perform the initial sketching phase of the **singular value decomposition (SVD)**, thereby eliminating the need for computationally expensive floating point operations. This adaptation not only speeds

up the computation but also decreases the memory overhead by avoiding the overestimation of matrix ranks—a common issue that leads to significant runtime and storage penalties.

Moreover, our exploration into optimized **QR decomposition methods**, augmented by recent advancements in parallel processing and randomized numerical techniques, has led to a further reduction in computational complexity. By incorporating a factor γ, where $0 < \gamma < 1$ represents the efficiency gain from these optimized QR methods, we have recalibrated the complexity of the orthogonalization step from $O(mk^2)$ to $O(\gamma mk^2)$. This adjustment has a cascading effect on the overall runtime of the Randomized Rangefinder within the GaLore algorithm, now effectively reducing it to:

$$O(mn \log n + r^3 + \gamma mnr + \gamma mr^2 + r^2 n)$$

This formulation not only underscores the impact of γ but also highlights our algorithm's independence from traditional hyperparameters such as the predetermined rank r, relying instead on dynamically determined numerical ranks.

In conclusion, this paper extends the theoretical and practical horizons of memory-efficient algorithms in machine learning. By integrating theoretical advancements with practical algorithm design, we provide a robust framework that supports more extensive and complex data analyses on constrained hardware environments. The implications of this work are broad, offering potential applications in areas ranging from neural network training to complex system simulations where efficiency and scalability are paramount.

Future work will explore further enhancements to the random sign matrix approach, seeking even greater reductions in computational overhead, and expanding the adaptability of the GaLore algorithm to a wider array of function types beyond those that are strictly reversible. Additionally, an exploration into the theoretical bounds of γ and its potential impacts on algorithmic stability and performance will be essential for refining our understanding of these advanced computational techniques.

References

[1] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection, 2024.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[3] Maike Meier and Yuji Nakatsukasa. Fast randomized numerical rank estimation for numerically low-rank matrices, 2024.

[4] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2010.

[5] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 2001. URL `http://www.yaroslavvb.com/papers/achlioptas-database.pdf`.

[6] Mark Rudelson and Roman Vershynin. Non-asymptotic theory of random matrices: extreme singular values, 2010.

[7] Avner Magen and Anastasios Zouzias. Low rank matrix-valued chernoff bounds and approximate matrix multiplication, 2010.

[8] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26(189-206):1, 1984.

[9] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *arXiv preprint arXiv:0909.4061*, 2009.

[10] Per-Gunnar Martinsson. Randomized numerical linear algebra: Foundations & algorithms. *arXiv preprint arXiv:2002.01387*, 2020.

[11] James Demmel, Laura Grigori, Mark Hoemmen, and Julien Langou. Communication-optimal parallel and sequential qr and lu factorizations. *SIAM Journal on Scientific Computing*, 34(1):A206–A239, 2012.

[12] Gregorio Quintana-Ortí, Xiaobai Sun, and Christian H. Bischof. Improving the performance of parallel qr factorization. *Parallel Computing*, 24(4):693–708, 1998.