Automatic Differentiation of Sketched Regression

Hang Liao[a] **Barak A. Pearlmutter**[*] **Vamsi K. Potluru**[†] **David P. Woodruff**[a]
CMU[a] NUIM[*] Comcast Research[†]

Abstract

Sketching for speeding up regression problems involves using a sketching matrix S to quickly find the approximate solution to a linear least squares regression (LLS) problem: given A of size $n \times d$, with $n \gg d$, along with b of size $n \times 1$, we seek a vector y with minimal regression error $\|Ay - b\|_2$. This approximation technique is now standard in data science, and many software systems use sketched regression internally, as a component. It is often useful to calculate derivatives (gradients for the purpose of optimization, for example) of such large systems, where sketched LLS is merely a component of a larger system whose derivatives are needed. To support Automatic Differentiation (AD) of systems containing sketched LLS, we consider propagating derivatives through LLS: both propagating perturbations (forward AD) and gradients (reverse AD). AD performs accurate differentiation and is efficient for problems with a huge number of independent variables. Since we use LLS_S (sketched LLS) instead of LLS for reasons of efficiency, propagation of derivatives also needs to trade accuracy for efficiency, presumably by sketching. There are two approaches for this: **(a)** use AD to transform the code that defines LLS_S, or **(b)** approximate exact derivative propagation through LLS using sketching methods. We provide strong bounds on the errors produced due to these two natural forms of sketching in the context of AD, giving the first dimensionality reduction analysis for calculating the derivatives of a sketched computation. Our results crucially depend on a novel analysis of the operator norm of a sketched inverse matrix product in this context. Extensive experiments on both synthetic and real-world experiments demonstrate the efficacy of our sketched gradients.

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

1 Introduction

Linear least-squares regression (LLS) is one of the oldest tool in the data scientist's toolkit, dating back to Gauss. Modern systems support LLS in libraries, with many variants for trading speed vs accuracy, for various kinds of sparse data, etc. Just as an FFT routine might be called from deep inside a larger program, LLS is often performed as a single step inside a much larger computational process. Deep learning, i.e., differentiable programming, systems are evolving to allow gradients and other derivatives to be automatically computed more freely and more efficiently: for instance, to allow complicated procedures invoking many library routines to be automatically differentiated, and to allow nesting of differentiation. For this program to come to fruition, we must know how to efficiently calculate both forward- and reverse-mode derivatives of all available numeric library functions. Here, we are concerned with how to perform AD on a system which internally uses sketched LLS. We first review linear regression and its sketched approximation, and the favorable complexity and accuracy bounds which sketched LLS provides. We then proceed to explore how the derivative (forward and reverse) of sketched LLS can be expressed. The naïve way of taking these derivatives results in poor error bounds. We therefore explore more sophisticated ways to calculate these derivatives, using a novel independently-sketched approximation to the derivative of a sketched LLS, and derive favorable error bounds. An expository example is presented in which sketched LLS is used as a routine within a larger system, and joint gradient optimization of the entire process is performed, requiring AD of the sketched LLS.

1.1 Linear Regression

Consider the classical problem of linear least-squares regression: find $x \in \mathbb{R}^d$ for which Ax is as close as possible to $b \in \mathbb{R}^n$, with $A \in \mathbb{R}^{n \times d}$ and $n > d$. This is the "tall and skinny" case, where A has more rows than columns (Seal, 1967). Let LLS denote the function that performs this mapping:

$$y = \mathrm{LLS}(A, b) \equiv \operatorname*{argmin}_x \|Ax - b\|_2^2 \tag{1}$$

Since there are more equations than variables, it is likely that there is no solution with objective value zero and hence we settle for one which is closest to b within the span of the matrix A. There is an analytic solution

$$y = A^\dagger b = (A^\mathrm{T} A)^{-1} A^\mathrm{T} b = \mathrm{LS}(A^\mathrm{T} A, A^\mathrm{T} b) \tag{2}$$

where $A^\dagger$ is the Moore-Penrose pseudo-inverse of A (Moore, 1920; Penrose, 1956) and $\mathrm{LS} : (M, b) \mapsto M^{-1} b$ is a linear system solver. Note that it is generally faster to implement LLS using a special-purpose iterative linear least-squares solver. We will treat LS and LLS as black boxes, ignoring their implementation details.

1.2 Sketched Linear Least Squares Regression

Oblivious subspace embeddings (OSE) generalize classical embedding results from vectors to subspaces. They are now widely used to speed up the regression problem, at the expense of accuracy (for a survey, see Woodruff, 2014). An OSE is a distribution on matrices $S \in \mathbb{R}^{m \times n}$ with $m \ll n$ such that for any given d-dimensional subspace U, with large probability S preserves the norm of every vector in U. More formally, we require that with probability at least $1 - \delta$:

$$\|Sx\| = (1 \pm \epsilon)\|x\| \qquad \forall x \in U, \tag{3}$$

where for a vector z, $\|z\|$ denotes the 2-norm $(\sum_i z_i^2)^{1/2}$. We can solve a linear system by sketching to approximately preserve the linear span of the columns in A, which span a d-dimensional subspace of $\mathbb{R}^n$. By this, we mean the following application of OSEs to least squares regression: we use a matrix S as follows:

$$y_S = \mathrm{LLS}_S(A, b) \equiv \mathrm{LLS}(SA, Sb) = \mathrm{LS}(M_S, m_S) \tag{4}$$

where $M_S \equiv (SA)^\mathrm{T} SA$ and $m_S \equiv (SA)^\mathrm{T} Sb$. Note we have replaced the original $n \times d$ problem with an $m \times d$ one, which serves as an approximation to the original problem. Since we will choose m such that $d \leq m \ll n$, this can significantly improve the running time. In particular for the case where S is the Subsampled Randomized Hadamard transform (SRHT) (Woodruff, 2014), we can compute SA in $O(nd \log n)$ time and the overall running time is $O(nd \log n + d^\omega/\epsilon^2)$, where ω is the exponent of matrix multiplication. One can also use CountSketch or a Gaussian families of matrices, see, e.g., Woodruff (2014) for a survey.

1.3 Algorithmic Differentiation

Algorithmic differentiation (AD) is a process by which a numerical calculation specified in a computer programming language can be mechanically transformed so as to calculate derivatives (in the differential calculus sense) of the function originally calculated (Wengert, 1964; Speelpenning, 1980; Rall, 1981; Griewank and Walther, 2008). We consider the two fundamental AD transformations: forward accumulation and reverse accumulation. In building deep learning frameworks, it is necessary for each operation (meaning operations like multiplication and addition, and also array operations regarded as primitives like convolution or matrix product) to have associated derivative propagation functions (a compilation of these for many standard matrix functions can be found in Giles, 2008). These come in two varieties: those that map a perturbation of the inputs to a basis function to a perturbation of its outputs, as needed for a forward AD transform; and those that map a gradient with respect to the outputs of a basis function to a gradient with respect to its inputs, as needed for a reverse AD transform (Naumann, 2012). We consider the least squares regression function which is computationally expensive, and is instead approximated using sketching methods (see, e.g., (Woodruff, 2014) for a survey). We study the propagation of derivatives through such computations, with attention to both computational burden and error.

Since derivative-taking and approximation do not commute (Sirkes and Tziperman, 1997), the straightforward strategy of taking the exact derivative of the sketched computation may yield numerically poor results. Moreover, in practice, the software systems to which this work is most immediately relevant to involve computing stochastic gradients via reverse AD using small subsets of a large dataset. Thus, introducing zero-mean error into the gradient calculation is usually not a serious concern, whereas introducing non-zero-mean (i.e., biased) error might derail the optimization.

We therefore search for methods to approximately propagate gradients through these burdensome computations which are both computationally efficient and unbiased. We establish strong bounds on the variance of the error, and also examine the expectation of the error with the intent of finding sufficient conditions for

the expectation of the error in gradient propagation to be zero.

1.4 Our Contributions

We revisit the error bound of the solution of sketched LLS and analyze the forward accumulation and reverse accumulation of least squares linear regression when combined with sketching and in particular we:

- prove a novel bound on the operator norm of $X = (SA)^\dagger SB$ and show that $\|X\|_2 = \|(SA)^\dagger SB\|_2 \leq \|A^\dagger B\|_2 + \epsilon\sqrt{(1+d/k)(\|B\|_2^2 + \|B\|_F^2/k)})/\sigma_{\min}(A)$, where S has $\tilde{O}(k/\epsilon^2)$[1] rows, Σ is the matrix in A's singular value decomposition $U\Sigma V^T$ and $\sigma_{\min}(A)$ is the minimum singular value of A. This extends the previously well-known result for vectors (Sarlos, 2006; Drineas et al., 2011).
- propose two ways of combining sketching with regression problems: (a) "Sketch + Differentiate": sketch the regression problem, as is usually done in the literature, and apply standard AD transformations for the forward mode and reverse mode. (b) "Differentiate + Sketch": obtain standard AD transformations on the original regression problem and only sketch the computationally intensive terms such as $A^{\mathrm{T}}A$. Empirically, the latter approach provides a better approximation of the unsketched calculation.

2 Main Results

One of the terms that shows up in the analysis of AD is the operator norm of $A^\dagger B$.

We start with the following definition.

Definition 1. *An (ϵ, δ, r, l)-oblivious subspace embedding is a distribution D over $\mathbb{R}^{m\times n}$ for which*

$$E_{S\sim D}\|U^{\mathrm{T}}S^{\mathrm{T}}SU - I\|^l < \epsilon^l\delta$$

where $\forall U \in \mathbb{R}^{n\times r}$, $U^{\mathrm{T}}U = I$.

We need the following lemma.

Lemma 2. *Let A be a full column rank matrix of size $n \times d$ and similarly for B, where we assume $\log(n) = d^{o(1)}$. Let S be an SRHT with $m = \tilde{O}((d + \log(1/\delta))/\epsilon^2)$ rows. For any matrix B of size $n \times d$ we have*

$\|X\|_2 = \|(SA)^\dagger SB\|_2 \leq \|A^\dagger B\|_2 + \epsilon\sqrt{(1+d/k)(\|B\|_2^2 + \|B\|_F^2/k)})/\sigma_{min}(A)$, with probability $1 - 1/poly(d)$.

[1]For a function f, we use the notation $\tilde{O}(f)$ to denote $f \cdot \mathrm{polylog}(f)$.

Proof. Letting $A = U\Sigma V^T$ be its singular value decomposition, where U and V have orthonormal columns and Σ is the non-negative diagonal matrix of singular values, we can expand X as follows:

$$\begin{aligned}(SA)^\dagger SB &= (A^{\mathrm{T}}S^{\mathrm{T}}SA)^{-1}A^{\mathrm{T}}S^{\mathrm{T}}SB \\ &= V\Sigma^{-1}(U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1}U^{\mathrm{T}}S^{\mathrm{T}}SB \\ &= V\Sigma^{-1}(I_d - T)^{-1}U^{\mathrm{T}}S^{\mathrm{T}}SB \\ &= V\Sigma^{-1}(\sum_{k=0}^{\infty}T^k)U^{\mathrm{T}}S^{\mathrm{T}}SB \qquad (5)\end{aligned}$$

where $T = I_d - U^{\mathrm{T}}S^{\mathrm{T}}SU$ and we have that $\|T\|_2 < 1$ for $\epsilon < 1$. Hence, the infinite von Neumann series converges to $(I_d - T)^{-1}$.

We also need the following result concerning approximate matrix product (AMM) with respect to the spectral norm (Cohen et al., 2016):

Theorem 3. *(Spectral Norm Approximate Matrix Product) For any $(\epsilon, \delta, 2k)$-OSE, it satisfies (k, ϵ, δ)-AMM, meaning the following condition holds:*

$$\begin{aligned}&\|A^{\mathrm{T}}S^{\mathrm{T}}SB - A^{\mathrm{T}}B\|_2 \\ &\qquad \leq \epsilon\sqrt{(\|A\|_2^2 + \|A\|_F^2/k)(\|B\|_2^2 + \|B\|_F^2/k)} \qquad (6)\end{aligned}$$

where k is the maximum stable rank of the matrices A, B with probability $1 - \delta$.

Continuing our analysis,

$$\begin{aligned}&\|V\Sigma^{-1}T^iU^{\mathrm{T}}S^{\mathrm{T}}SB\|_2 \leq \epsilon^i\|\Sigma^{-1}\|_2\|U^{\mathrm{T}}S^{\mathrm{T}}SB\|_2 \\ &\sum_{i=0}^{\infty}\|V\Sigma^{-1}T^iU^{\mathrm{T}}S^{\mathrm{T}}SB\| \leq \frac{1}{1-\epsilon}\|\Sigma^{-1}\|_2\|U^{\mathrm{T}}S^{\mathrm{T}}SB\|_2 \\ &\qquad \leq (1+2\epsilon)\|\Sigma^{-1}\|_2\|U^{\mathrm{T}}S^{\mathrm{T}}SB\|_2. \qquad (7)\end{aligned}$$

Here we use the fact (see, e.g., Woodruff, 2014) that with probability $1-\exp(-d)$, the singular values of SU are in $(1 \pm \epsilon)$ for $m = O(d/\epsilon^2)$. Applying Theorem 3 and upper bounding the stable rank by the rank,

$$\begin{aligned}&\|U^{\mathrm{T}}S^{\mathrm{T}}SB - U^{\mathrm{T}}B\|_2 \\ &\qquad \leq \epsilon\sqrt{(\|U\|_2^2 + \|U\|_F^2/k)(\|B\|_2^2 + \|B\|_F^2/k)} \\ &\qquad \leq \epsilon\sqrt{(1+d/k)(\|B\|_2^2 + \|B\|_F^2/k)} \qquad (8)\end{aligned}$$

where we used that the spectral norm of a matrix does not change when a rotation matrix such as U is applied. Since we only utilized subspace embedding results as a black box box, our result is valid for sub-Gaussian maps, SRHT, or sparse subspace embeddings with a suitable choice on the number of rows of the sketching matrix S. We refer the reader to Woodruff

(2014) for the exact bounds; we note that in each case the number of rows of S is $\text{poly}(d/\epsilon)$ and the different sketching matrices have different properties; the precise bound of the SRHT in the lemma statement follows. □

We use the result above to bound the following term where $y = \text{argmin}_x \|Ax - b\|$ and $y_S = \text{argmin}_x \|SAx - Sb\|$. We have that with probability $1 - 1/\text{poly}(d)$,

$$\begin{aligned}
&\|M^{-1}A^TBy - M_S^{-1}A^TS^TSBy_s\|_2 \\
&\quad = \|M^{-1}A^TBy - M_S^{-1}A^TS^TSB(y_S - y) \\
&\qquad - M_S^{-1}A^TS^TSBy\|_2 \\
&\quad \leq \|M_S^{-1}A^TS^TSB(y_S - y)\|_2 \\
&\qquad + \|M^{-1}A^TBy - M_S^{-1}A^TS^TSBy\|_2 \\
&\quad \leq \|M_S^{-1}A^TS^TSB\|_2\|y_S - y\|_2 \\
&\qquad + O(\epsilon)\min_x \|Ax - By\|/\sigma_{\min}(A) \\
&\quad \leq O(\epsilon)(\|X\|_2\|Ay - b\|_2/\sigma_{\min}(A) \\
&\qquad + \min_x \|Ax - By\|_2/\sigma_{\min}(A)) \qquad (9)
\end{aligned}$$

where $X = \text{argmin}_X \|SAX - SB\|_F$ and we used the result of Sarlos (2006) for individual vectors that $\|y_s - y\| \leq O(\epsilon)\|Ay - b\|_2\|A^\dagger\|_2$ with $\|A^\dagger\|_2 = 1/\sigma_{\min}(A)$ with probability $1 - 1/\text{poly}(d)$, applying it to both the first and second terms corresponding to the two least squares problems.

3 Automatic Differentiation (AD)

Automatic differentiation is a process by which a numerical calculation specified in a computer programming language can be mechanically transformed so as to calculate derivatives (in the differential calculus sense) of the function originally calculated (Wengert, 1964; Speelpenning, 1980; Rall, 1981; Griewank and Walther, 2008). If $f : (x \in \mathbb{R}^n) \mapsto (y \in \mathbb{R}^m)$ and $J_{f(x)} \in \mathbb{R}^{m \times n}$ is the Jacobian of f at x (the matrix of partial derivatives $\partial y_i/\partial x_j$), then these are

$$\mathrm{d}y = J_{f(x)}\mathrm{d}x \qquad \nabla_x E = J_{f(x)}{}^{\mathrm{T}}\nabla_y E \qquad (10)$$

respectively. Note that E is the objective function of the downstream application, for example, the logistic loss function for a classification task. When we generalize to inputs and outputs that are not just vectors of real numbers (for instance, the input to a linear system solver would have shape $(\mathbb{R}^{n\times n}, \mathbb{R}^n)$, that is, a pair containing an $n \times n$ array and an n-dimensional vector), by analogy we use $\mathcal{J}$ for operators involving the generalized Jacobian:

$$\mathrm{d}y = \mathcal{J}\, f(x)(\mathrm{d}x) \qquad \nabla_x E = \mathcal{J}^{\mathrm{T}}\, f(x)(\nabla_y E) \qquad (11)$$

Note that, like multiplication by the Jacobian, these derivative propagations are by definition linear, since they are defined as the local linear approximation to f at x, and its adjoint. This implies a relationship between $\mathcal{J}$ and $\mathcal{J}^{\mathrm{T}}$, namely, for any $\mathrm{d}x$ and $\nabla_y E$,

$$\langle \mathcal{J}^{\mathrm{T}}\, f(x)(\nabla_y E), \mathrm{d}x\rangle = \langle \nabla_y E, \mathcal{J}\, f(x)(\mathrm{d}x)\rangle \qquad (12)$$

where $\langle\cdot,\cdot\rangle$ is the appropriately generalized dot product.

3.1 Notation

We use $\dot{x} = \mathrm{d}x$ for an infinitesimal perturbation of x (technically $\dot{x} \in T_x\alpha$ is an element of the tangent space of α at $x \in \alpha$) and $\bar{x} = \nabla_x E$ for a sensitivity with respect to x (technically $\bar{x} \in T_x^*\alpha$ is an element of the co-tangent space of α at x). For uniformity, we adopt this standard notation used in the AD community. We use a subscript S for sketched computations and values thereby produced: $y_S = f_S(x)$. We use $\dot{y} = \mathcal{J}\, f(x)(\dot{x})$ to denote the function that propagates forward derivatives through f at x, and $\bar{x} = \mathcal{J}^{\mathrm{T}}\, f(x)(\bar{y})$ for the function that propagates reverse derivatives through f at x. When there are multiple sketched computations involving different sketching matrices drawn independently these are distinguished with S, S', S'', etc. We write $u' = (c \pm \epsilon)u$ for $(c - \epsilon)u \leq u' \leq (c + \epsilon)u$.

3.2 AD for regression

Consider solving a linear system of equations $Mz = m$. We write $z = \text{LS}(M, m)$ to clearly denote the input (M and m) and output (z) of the process, and to give the process itself a name (LS). For the forward accumulation mode AD transform of this process we write $(z, \dot{z}) = \mathcal{J}\,\text{LS}(M, m)(\dot{M}, \dot{m})$ where the mapping from the perturbation of the inputs to the perturbations of the outputs is $\dot{z} = \text{LS}(M, \dot{m} - \dot{M}z)$. In the case of linear regression, we have $M = A^{\mathrm{T}}A$ and $m = A^{\mathrm{T}}b$. The forward accumulation is given by: $(\dot{M}, \dot{m}) = (\dot{A}^{\mathrm{T}}A + A^{\mathrm{T}}\dot{A}, \dot{A}^{\mathrm{T}}b + A^{\mathrm{T}}\dot{b})$. The reverse AD transform for a linear system solver is $(\bar{M}, \bar{m}) = \mathcal{J}^{\mathrm{T}}\,\text{LS}(M, m)(\bar{z}) = (-\bar{m}z^{\mathrm{T}}, \text{LS}(M^{\mathrm{T}}, \bar{z}))$. Note that this involves solving a transposed system of equations.

Similarly, the transform of matrix multiplication $Z = XY = (\times)(X, Y)$ is $(\bar{X}, \bar{Y}) = \mathcal{J}^{\mathrm{T}}(\times)(X, Y)(\bar{Z}) = (\bar{Z}Y^{\mathrm{T}}, X^{\mathrm{T}}\bar{Z})$ and the transform of $Z = X^{\mathrm{T}} = (\cdot^{\mathrm{T}})(X)$ is trivial: $\bar{X} = \mathcal{J}^{\mathrm{T}}(\cdot^{\mathrm{T}})(X)(\bar{Z}) = \bar{Z}^{\mathrm{T}}$.

Lemma 4. *If S is a sketching matrix of size $m \times n$ where $m = O(d/\epsilon^2)$ and $n \leq$ poly(d), then with probability $1 - 1/$poly(d), we have $\|AM^{-1} - AM_S^{-1}\|_F \leq \epsilon\|\Sigma^{-1}\|_F$, $\|M^{-1} + M_S^{-1}\|_2 \leq (2 + \epsilon)\|\Sigma^{-1}\|_2^2$ and $\|M^{-1} - M_S^{-1}\|_2 \leq \epsilon\|\Sigma^{-1}\|_2\|\Sigma^{-1}\|_F$ where the singular values of SU are in the range $[1 - \epsilon, 1 + \epsilon]$.*

Proof. Consider the SVD of the matrix A. We have $M = A^{\mathrm{T}}A = V\Sigma U^{\mathrm{T}}U\Sigma V^{\mathrm{T}} = V\Sigma^2 V^{\mathrm{T}}$ and $M^{-1} = V\Sigma^{-2}V^{\mathrm{T}}$. Additionally, $AM^{-1} = U\Sigma V^{\mathrm{T}}V\Sigma^{-2}V^{\mathrm{T}} = U\Sigma^{-1}V^{\mathrm{T}}$ yielding $\|AM^{-1}\|_2 = \|\Sigma^{-1}\|_2$. So, we can simplify the expression $AM_S^{-1} = U\Sigma V^{\mathrm{T}}(V\Sigma U^{\mathrm{T}}S^{\mathrm{T}}SU\Sigma V^{\mathrm{T}})^{-1}$ and obtain $U\Sigma V^{\mathrm{T}}V\Sigma^{-1}(U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1}\Sigma^{-1}V^{\mathrm{T}} = U(U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1}\Sigma^{-1}V^{\mathrm{T}}$ and bound it $\|AM_S^{-1}\|_2 = (1\pm\epsilon)\|\Sigma^{-1}\|_2$ with probability at least $1 - 1/\mathrm{poly}(d)$. This enables us to prove the required bounds:

$$\begin{aligned}\|M^{-1} - M_S^{-1}\|_F &= \|V\Sigma^{-1}(I - (U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1})\Sigma^{-1}V^{\mathrm{T}}\| \\ &\le \|\Sigma^{-1}\|_2\|I - (U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1})\|_2\|\Sigma^{-1}\| \\ &\le \epsilon\|\Sigma^{-1}\|_2\|\Sigma^{-1}\|_F \qquad (13)\\ \|M^{-1} + M_S^{-1}\|_2 &= \|V\Sigma^{-1}(I + (U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1})\Sigma^{-1}V^{\mathrm{T}}\| \\ &\le \|\Sigma^{-1}\|_2\|I + (U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1})\|_2\|\Sigma^{-1}\|_2 \\ &\le (2+\epsilon)\|\Sigma^{-1}\|_2^2\end{aligned}$$

We thus have:

$$\begin{aligned}\|AM^{-1} - AM_S^{-1}\|_F &= \|U(I - (U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1})\Sigma^{-1}V^{\mathrm{T}}\| \\ &= \|(I - (U^{\mathrm{T}}S^{\mathrm{T}}SU)^{-1})\|_2\|\Sigma^{-1}V^{\mathrm{T}}\|_F \\ &\le \epsilon\|\Sigma^{-1}\|_F\end{aligned}$$

where we used $(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$. □

4 Forward Mode AD

We have $A^{\mathrm{T}}Ay = A^{\mathrm{T}}b$ for the least squares linear regression problem. Expanding out the terms by perturbing the input and dropping the second order terms we obtain $A^{\mathrm{T}}A\dot{y} = \dot{A}^{\mathrm{T}}b + A^{\mathrm{T}}\dot{b} - (\dot{A}^{\mathrm{T}}A + A^{\mathrm{T}}\dot{A})y$. (AD transforms for a few important functions can be found in the Supplementary Material.) We consider the following two ways to obtain efficient versions of the forward mode by employing sketching:

4.1 Sketch and Differentiate

Let us denote the solution of the sketched linear regression problem by y_s. So, we have $A^{\mathrm{T}}S^{\mathrm{T}}SA\dot{y}_S = \dot{A}^{\mathrm{T}}S^{\mathrm{T}}Sb + A^{\mathrm{T}}S^{\mathrm{T}}S\dot{b} - (\dot{A}^{\mathrm{T}}S^{\mathrm{T}}SA + A^{\mathrm{T}}S^{\mathrm{T}}S\dot{A})y_S$ where $y_S = \mathrm{LS}(A^{\mathrm{T}}S^{\mathrm{T}}SA, A^{\mathrm{T}}S^{\mathrm{T}}Sb)$.

Lemma 5. *Given matrix S satisfying the (ϵ,δ,d,l)-OSE moment property for some $l \ge 2$, we can bound $\|\dot{y} - \dot{y}_S\|_2$ with probability $1-\delta$ as follows:*

Proof.

$$\begin{aligned}\|\dot{y} - \dot{y}_S\|_2 &= \|M^{-1}(\dot{A}^{\mathrm{T}}b + A^{\mathrm{T}}\dot{b} - (\dot{A}^{\mathrm{T}}A + A^{\mathrm{T}}\dot{A})y) \\ &\quad - M_S^{-1}(\dot{A}^{\mathrm{T}}S^{\mathrm{T}}Sb + A^{\mathrm{T}}S^{\mathrm{T}}S\dot{b} - (\dot{A}^{\mathrm{T}}S^{\mathrm{T}}SA \\ &\quad + A^{\mathrm{T}}S^{\mathrm{T}}S\dot{A})y_S)\|_2 \\ &\le \|M^{-1}\dot{A}^{\mathrm{T}}b - M_S^{-1}\dot{A}^{\mathrm{T}}S^{\mathrm{T}}Sb\|_2 \\ &\quad + \|M^{-1}A^{\mathrm{T}}\dot{b} - M_S^{-1}A^{\mathrm{T}}S^{\mathrm{T}}S\dot{b}\|_2 \\ &\quad + \|M^{-1}(\dot{A}^{\mathrm{T}}A + A^{\mathrm{T}}\dot{A})y \\ &\quad - M_S^{-1}(\dot{A}^{\mathrm{T}}S^{\mathrm{T}}SA + A^{\mathrm{T}}S^{\mathrm{T}}S\dot{A})y_S\|_2 \qquad (14)\end{aligned}$$

By the triangle inequality, $\|AB - CD\| \le \|A-C\|\|B+D\| + \|A+C\|\|B-D\|$ for arbitrary conforming matrices, which we will use in the following equations. Let us bound each of the differences between a term and its sketched version. The first difference term can be written as

$$\begin{aligned}&\|M^{-1}\dot{A}^{\mathrm{T}}b - M_S^{-1}\dot{A}^{\mathrm{T}}S^{\mathrm{T}}Sb\| \\ &\quad\le \|M^{-1} - M_S^{-1}\|\|\dot{A}^{\mathrm{T}}b + \dot{A}^{\mathrm{T}}S^{\mathrm{T}}Sb\| \\ &\quad + \|M^{-1} + M_S^{-1}\|_2\|\dot{A}^{\mathrm{T}}b - \dot{A}^{\mathrm{T}}S^{\mathrm{T}}Sb\| \\ &\quad\le O(\epsilon)\|\Sigma^{-1}\|_2\|\Sigma^{-1}\|_F\|\dot{A}^{\mathrm{T}}b + \dot{A}^{\mathrm{T}}S^{\mathrm{T}}Sb\| \\ &\quad + (2+\epsilon)\|\Sigma^{-1}\|_2^2\|\dot{A}^{\mathrm{T}}b - \dot{A}^{\mathrm{T}}S^{\mathrm{T}}Sb\|\end{aligned}$$

where we used Lemma 4 and that the singular values of SU are in $1\pm\epsilon$. Following up with the second difference term

$$\|M^{-1}A^{\mathrm{T}}\dot{b} - M_S^{-1}A^{\mathrm{T}}S^{\mathrm{T}}S\dot{b}\| \le O(\epsilon)\min_x\|Ax - \dot{b}\|_2\|A^{\dagger}\|_2 \qquad (15)$$

where we used the result of Price et al. (2017). The third difference term can be handled as follows:

$$\begin{aligned}&\|M^{-1}\dot{A}^{\mathrm{T}}Ay - M_S^{-1}\dot{A}^{\mathrm{T}}S^{\mathrm{T}}SAy_S\|_2 \\ &\le \|M^{-1} - M_S^{-1}\|_2\|\dot{A}^{\mathrm{T}}Ay + \dot{A}^{\mathrm{T}}S^{\mathrm{T}}SAy_S\|_2 \qquad (16)\\ &\quad + \|M^{-1} + M_S^{-1}\|_2\|\dot{A}^{\mathrm{T}}Ay - \dot{A}^{\mathrm{T}}S^{\mathrm{T}}SAy_S\|_2 \\ &\le \epsilon\|\Sigma^{-1}\|_2\|\Sigma^{-1}\|_F\|\dot{A}^{\mathrm{T}}Ay + \dot{A}^{\mathrm{T}}S^{\mathrm{T}}SAy_S\| \\ &\quad + (2+\epsilon)\|\Sigma^{-1}\|_2^2\|\dot{A}^{\mathrm{T}}Ay - \dot{A}^{\mathrm{T}}S^{\mathrm{T}}SAy_S\| \\ &\le \epsilon\|\Sigma^{-1}\|_2\|\Sigma^{-1}\|_F\|\dot{A}^{\mathrm{T}}Ay + \dot{A}^{\mathrm{T}}S^{\mathrm{T}}SAy_S\| \\ &\quad + (2+\epsilon)\|\Sigma^{-1}\|_2^2(\|\dot{A}^{\mathrm{T}}A\|\|y - y_S\| \\ &\quad + \|\dot{A}^{\mathrm{T}}A - \dot{A}^{\mathrm{T}}S^{\mathrm{T}}SA\|_2\|y_S\|_2)\end{aligned}$$

where we used Lemma 4 and we can apply AMM (Cohen et al., 2016) to bound the spectral norm of the matrix product $\dot{A}^{\mathrm{T}}A$ along with the solution bound for $\|y - y_S\|$ (Price et al., 2017). Note that the bounds go to zero when S is the identity matrix. The last difference term can be bounded from the result in Sec-

tion 2 as follows:

$$\begin{aligned}
&\|M^{-1}A^{\mathrm{T}}\dot{A}y - M_S^{-1}A^{\mathrm{T}}S^{\mathrm{T}}S\dot{A}y_S\| \\
&\quad \le O(\epsilon)\frac{\|X\|_2}{\sigma_{\min}(A)}\|Ay-b\|_2 + \min_x \frac{\|Ax-\dot{A}y\|_2}{\sigma_{\min}(A))} \\
&\|X\|_2 \lesssim (1+2\epsilon)\|\Sigma^{-1}\|_2 \\
&\quad (\|\dot{A}\|_2 + \epsilon\sqrt{(1+d/k)(\|\dot{A}\|_2^2 + \|\dot{A}\|_F^2/k)})
\end{aligned}$$

Combining all of these results for the four terms gives us the required approximation bound. □

4.2 Differentiate and Sketch

Let us denote the solution of the least squares problem by y_D, where we sketch only the computationally expensive term M as follows: $y_D = \mathrm{LS}(A^{\mathrm{T}}S^{\mathrm{T}}SA, A^{\mathrm{T}}b)$. A similar consideration of only sketching the matrix A was first considered in the setting of constrained least-squares Pilanci and Wainwright (2016). For ease of exposition, denote $G = \dot{A}^{\mathrm{T}}A + A^{\mathrm{T}}\dot{A}$.

Lemma 6. *For any matrix S satisfying the (ϵ, δ, d, l)-OSE moment property for some $l \ge 2$, we have with probability $1-\delta$*

$$\begin{aligned}
\|\dot{y} - \dot{y}_D\| \le O(\epsilon)(&\|\Sigma^{-1}\|_2^2\|\dot{A}^{\mathrm{T}}b \\
&+ A^{\mathrm{T}}\dot{b}\| + \|\Sigma^{-1}\|_2^2\|Gy\| + \|M_S^{-1}G\|\|A^{\mathrm{T}}b\|)
\end{aligned}$$

Proof. We only sketch the computationally intensive part corresponding to $A^{\mathrm{T}}A$. Let $A^{\mathrm{T}}S^{\mathrm{T}}SAy_D = A^{\mathrm{T}}b$ and $A^{\mathrm{T}}S^{\mathrm{T}}SA\dot{y}_D = \dot{A}^{\mathrm{T}}b + A^{\mathrm{T}}\dot{b} - (\dot{A}^{\mathrm{T}}A + A^{\mathrm{T}}\dot{A})y_D$. We bound $\|\dot{y} - \dot{y}_D\|$ using the results in Lemma 4:

$$\begin{aligned}
\|\dot{y} - \dot{y}_D\| &= \|(M^{-1} - M_S^{-1})(\dot{A}^{\mathrm{T}}b + A^{\mathrm{T}}\dot{b}) \\
&\quad + M_S^{-1}(\dot{A}^{\mathrm{T}}A + A^{\mathrm{T}}\dot{A})y_D \\
&\quad - M^{-1}(\dot{A}^{\mathrm{T}}A + A^{\mathrm{T}}\dot{A})y)\| \\
&\le \epsilon\|\Sigma^{-1}\|_2^2\|\dot{A}^{\mathrm{T}}b + A^{\mathrm{T}}\dot{b}\| \\
&\quad + \|(M^{-1} - M_S^{-1})Gy\| \\
&\quad + \|M_S^{-1}G(y - y_D)\| \\
&\le \epsilon\|\Sigma^{-1}\|_2^2\|\dot{A}^{\mathrm{T}}b + A^{\mathrm{T}}\dot{b}\| \\
&\quad + \epsilon\|\Sigma^{-1}\|_2^2\|Gy\| + \|M_S^{-1}G(y - y_D)\| \\
&\le \epsilon\|\Sigma^{-1}\|_2^2\|\dot{A}^{\mathrm{T}}b + A^{\mathrm{T}}\dot{b}\| \\
&\quad + \epsilon\|\Sigma^{-1}\|_2^2\|Gy\| + \epsilon\|M_S^{-1}G\|\|A^{\mathrm{T}}b\|
\end{aligned}$$

where the last term $\|y - y_D\|$ can be bounded as follows: $\|y - y_D\| = \|((A^{\mathrm{T}}A)^{-1} - (A^{\mathrm{T}}S^{\mathrm{T}}SA)^{-1})A^{\mathrm{T}}b\| \le \|((A^{\mathrm{T}}A)^{-1} - (A^{\mathrm{T}}S^{\mathrm{T}}SA)^{-1})\|_2\|A^{\mathrm{T}}b\|$. Since S is a subspace embedding for the column span of A, $(1-\epsilon)A^{\mathrm{T}}A \le A^{\mathrm{T}}S^{\mathrm{T}}SA \le (1+\epsilon)A^{\mathrm{T}}A$ in the PSD ordering, which implies $(1/(1+\epsilon))(A^{\mathrm{T}}A)^{-1} \le (A^{\mathrm{T}}S^{\mathrm{T}}SA)^{-1} \le (1/(1-\epsilon))(A^{\mathrm{T}}A)^{-1}$, which implies that $\|(A^{\mathrm{T}}A)^{-1} - (A^{\mathrm{T}}S^{\mathrm{T}}SA)^{-1}\|_2 = O(\epsilon)\|\Sigma^{-2}\|_2$. □

5 Reverse Mode AD

The sensitivities for the linear least squares problem are derived in the Supplementary Material, and in particular $(\bar{A}, \bar{b}) = (-A^{\dagger\mathrm{T}}\bar{y}y^{\mathrm{T}} - A\bar{y}\bar{y}^{\mathrm{T}}M^{-1} + b\bar{y}^{\mathrm{T}}M^{-1}, A^{\dagger T}\bar{y})$. Similar to the forward mode, we consider the following two approaches of "Sketch and Differentiate" and "Differentiate and Sketch". Applying this to the sketched regression problem, $M_S = (SA)^{\mathrm{T}}SA$, $m_S = (SA)^{\mathrm{T}}Sb$ and $y_S = \mathrm{LS}(M_S, m_s)$. In particular, we have $\bar{A}_S, \bar{b}_S = \mathcal{J}^{\mathrm{T}}(f_S)(S, A, b)(y_S, \bar{y})$. Whence, we have $\bar{A}_S = -S^{\mathrm{T}}A_S^{\dagger T}\bar{y}y_S{}^{\mathrm{T}} - S^{\mathrm{T}}SAy_S\bar{y}^{\mathrm{T}}M_S^{-1} + S^{\mathrm{T}}Sb\bar{y}^{\mathrm{T}}M_S^{-1}$. And also, $\bar{b}_S = S^{\mathrm{T}}SA\bar{m} = S^{\mathrm{T}}SAM_S{}^{-\mathrm{T}}\bar{y} = S^{\mathrm{T}}A_S^{\dagger T}\bar{y}$. Since it is hard to model the relationship between $\bar{y}$ and $\bar{y}_S$, we make an inaccurate but reasonable assumption that $\bar{y} = \bar{y}_S$.

5.1 "Sketch and Differentiate"

Lemma 7. *The reverse mode approximation error for the term $\bar{b}$ when we approximate it by the sketching matrix S can be bounded with probability $1-\delta$ as follows: $\|\bar{b} - \bar{b}_S\|_2 \le \|\Sigma^{-1}\|_2\|\bar{y}\|_2(\epsilon + (1+\epsilon)\|I - S^{\mathrm{T}}S\|_2)$.*

Proof. Use Lemma 4 and use sub-mulitplicativity. Hence, the error can be large ($\|I - S^{\mathrm{T}}S\|_2$). □

Lemma 8. *The reverse mode approximation error for the term $\bar{A}$ when we approximate it by the sketching matrix S can be bounded with probability $1-\delta$.*

Proof. The approximation error can be decomposed into three parts $\|\bar{A} - \bar{A}_S\| \le P_1 + P_2 + P_3$:

$$\begin{aligned}
P_1 &= \|b\bar{y}^{\mathrm{T}}M^{-1} - S^{\mathrm{T}}Sb\bar{y}_S{}^{\mathrm{T}}M_S^{-1})\|_F \\
&\le \|I - S^TS\|\|b\bar{y}^{\mathrm{T}}M_S^{-1}\| + \|b\bar{y}^{\mathrm{T}}(M^{-1} - M_S^{-1}\| \\
P_2 &\le \|AM^{-1}\bar{y}y^{\mathrm{T}} - AM_S^{-1}\bar{y}y^{\mathrm{T}}\|_F \\
&\quad + \|AM_S^{-1}\bar{y}y^{\mathrm{T}} - S^{\mathrm{T}}SAM_S^{-1}\bar{y}y_S{}^{\mathrm{T}}\|_F \\
&\le \epsilon\|\Sigma^{-1}\|_F\|\bar{y}\|\|y\| \\
&\quad + \|AM_S^{-1}\bar{y}y^{\mathrm{T}} - S^{\mathrm{T}}SAM_S^{-1}\bar{y}y_S{}^{\mathrm{T}}\|_F
\end{aligned}$$

We can similarly do the above for P_3. Also, note that the error for the terms P_1, P_2, P_3 can be large, similar to the previous lemma. □

5.2 "Differentiate and Sketch"

Lemma 9. *The reverse mode approximation error for the term $\bar{b}$ when we sketch only the computationally expensive terms by S, with probability at least $1-\delta$ satisfies $\|\bar{b} - \bar{b}_S\|_2 \le \epsilon\|\Sigma^{-1}\|_2\|\bar{y}\|_2$.*

Proof. We use the sketching properties and sub-multiplicativity, and the result follows. □

Lemma 10. *The reverse mode approximation error for the term $\bar{A}$, when we sketch only the computationally expensive terms by S, is, with probability at least $1-\delta$, bounded by $O(\epsilon)$.*

Proof. We give the proof in the Supplementary Material, and also state the exact form of the approximation error there. We utilize the result from Price et al. (2017) to bound the least squares solution error. □

6 Related work

Sketching for speeding up distributed communication of gradients has been recently considered (Ivkin et al., 2019, and references therein). In particular, distributed stochastic gradient descent (SGD) methods have been sped up by utilizing CountSketch projections. In our work, we explicitly open up the gradient computation step to identify parts that can be sped up by sketching methods. This marks a significant departure from the recent sketching for gradient literature which does not consider the structure of the layer when computing the gradient.

7 Experiments

We consider both synthetic and real-world datasets to highlight our regression layer and sketching for gradients to speed up training times. Synthetic experiments have been deferred to the Supplementary Material. The real-world datasets that we consider are the following:

MNIST: $60,000$ handwritten digits of shape 28×28 for training and $10,000$ for testing.

CIFAR10: $60,000$ images in 10 equal classes of which $10,000$ are for testing. The image classes include airplanes, horses, and cats.

Experimental results from the two approaches, namely, "sketch+differentiate" and "differentiate+sketch", are shown in Figures 1 and 2.

7.1 Autoencoder

Let us consider an autoencoder for showcasing our new regression layer. We consider the standard encode decoder framework with the encoder consisting of a linear layer mapping to 128 dimensions followed by a ReLu layer. The decoder is built with a linear layer mapping from 64 to 128 dimensions followed by a ReLU and a second linear layer mapping from 128 dimensions to the input dimension, followed by a tanh layer. In our experiments, we replace the only linear layer of the encoder by the regression layer or, in other words, the linear least squares regression module that has been considered in this paper.

Table 1: Tess loss on the MNIST and CIFAR10 datasets after convergence of the sketching algorithms using random Gaussian (RG) and CountSketch matrices (CS) with the two approaches proposed, namely *diffsketch* or **ds**, and *sketchdiff* or **sd**.

Sketching		MNIST			CIFAR10		
		64	128	256	64	128	256
RG	**ds**	0.16	0.08	0.08	0.21	0.09	0.08
	sd	0.11	0.07	0.08	0.11	0.08	0.08
CS	**ds**	0.15	0.10	0.09	0.14	0.09	0.09
	sd	0.10	0.09	0.08	0.09	0.08	0.08

7.2 Results

Overall, the regression layer tends to have a higher loss and running time compared with the linear layer. Yet, after applying sketching with Gaussian or CountSketch matrices, the running time is now close to the running time of a linear layer and not at the cost of a higher loss. Instead, we can see that with higher rank features $(128, 256)$, our sketched regression model achieves a lower loss and converges faster than the normal regression layer. The performance of the regression layer with CountSketch can be further improved with sparse datasets. We run our models on both a GPU and a CPU and find the performance of sketching algorithms to perform differently between the two. This is likely due to the fact that our sketching operations are not optimized for GPU settings, as seen in Figure 2.

8 Discussion and Future Work

We have shown that the advantages of sketched linear regression need not be sacrificed in the context of AD: in fact, sketching can be applied appropriately to the derivative computations in a fashion that preserves both the favorable complexity properties of AD and the favorable accuracy properties of sketching. One might speculate that this basic idea may hold for other sketched computations.

We provide two ways of speeding up a NN linear layer via sketching methods, and provide approximation bounds for the induced error. Experimental results are consistent with the theory, but do raise some tantalizing questions. For the example considered in

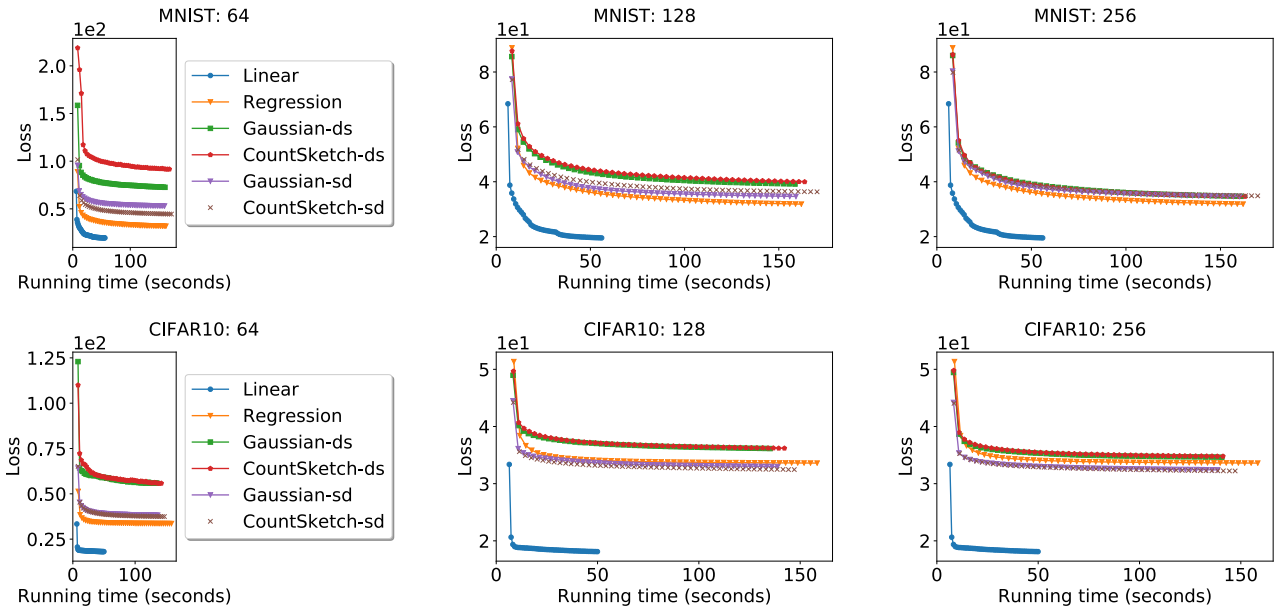

Figure 1: Training loss in linear and regression layers on MNIST and CIFAR10 datasets with 64, 128 and 256 rank features run on a NVIDIA GTX980 GPU. Notice that we hardly get any speedup over the plain regression layer and this is probably due to the fact that we have not taken advantage of GPU capabilities for implementing the sketching operations. Also, surprisingly the *sketchdiff* seems to result in better performance in terms of training loss than the *diffsketch* approach.

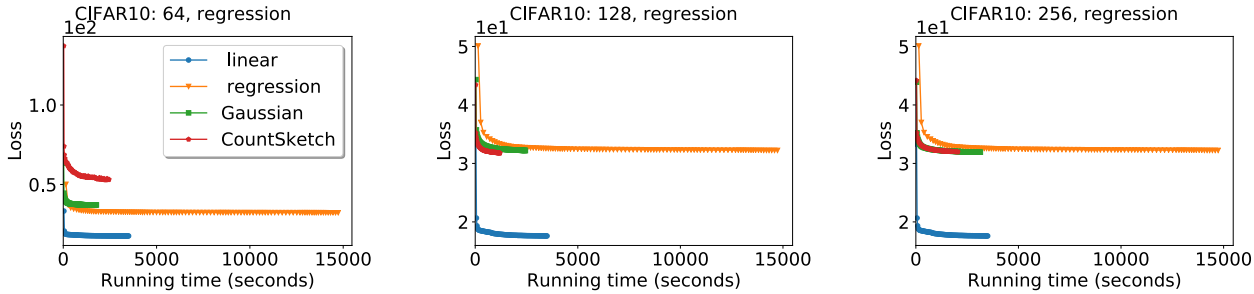

Figure 2: Training loss in linear and regression layers on CIFAR10 dataset with 64, 128 and 256 rank features on multicore settings (CPU). We note that sketching methods provide a significant speedup over the plain regression Layer. In the CPU setting, we only consider the *diffsketch* version though it should also apply to *sketchdiff*.

the Supplementary Material, *diffsketch* has a lower error compared to *sketchdiff*. However, with actual experiments with an autoencoder, the results appear to be reversed. We sketched only the linear layer in the encoder layer because the dimensions were favorable to our setting.

On the theoretical end, we have exhibited a useful bound on the spectral norm of $X = (SA)^{\dagger}SB$ which appears in the analysis of AD of sketched linear least squares regression. Also, two ways of approaching sketching are shown in the context of AD: "sketch and differentiate" vs "differentiate and sketch". Empirically, the former seems to have favorable approximation results which could help with selecting the appropriate approach when incorporating sketching tools into deep learning frameworks. Extensions of these results to other problems such as low-rank matrix approximation and regression with other norms would be of interest.

Acknowledgements

Vamsi P. is currently at JP Morgan AI Research. This paper was prepared for information purposes by the AI Research Group of JPMorgan Chase & Co

and its affiliates ("J.P. Morgan"), and is not a product of the Research Department of J.P. Morgan. D. Woodruff would like to thank partial support from the Office of Naval Research (ONR) grant N00014-18-1-2562.

References

Hilary L. Seal. The historical development of the Gauss linear model. *Biometrika*, 54(1/2):1–24, 1967. doi: 10.1093/biomet/54.1-2.1.

Eliakim Hastings Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26(9):394–5, 1920. doi: 10.1090/S0002-9904-1920-03322-7.

Roger Penrose. On best approximate solution of linear matrix equations. *Proceedings of the Cambridge Philosophical Society*, 52:17–9, 1956. doi: 10.1017/S0305004100030929.

David P Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

R. Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.

B. Speelpenning. *Compiling Fast Partial Derivatives of Functions Given by Algorithms*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, January 1980.

Louis B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer, Berlin, 1981. ISBN 0–540–10861–0. doi: 10.1007/3-540-10861-0.

Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 105 in Other Titles in Applied Mathematics. SIAM, Philadelphia, PA, 2nd edition, 2008. ISBN 978–0–898716–59–7. URL http://bookstore.siam.org/ot105/.

Mike B. Giles. Collected matrix derivative results for forward and reverse mode algorithmic differentiation. In Christian H. Bischof, H. Martin Bücker, Paul D. Hovland, Uwe Naumann, and J. Utke, editors, *Advances in Automatic Differentiation*, volume 64 of *Lecture Notes in Computational Science and Engineering*, pages 35–44. Springer, Berlin, 2008. ISBN 978-3-540-68935-5. doi: 10.1007/978-3-540-68942-3_4.

Uwe Naumann. *The Art of Differentiating Computer Programs: An Introduction to Algorithmic Differentiation*. Number 24 in Software, Environments, and Tools. SIAM, Philadelphia, PA, 2012. ISBN 978–1–611972–06–1. URL http://bookstore.siam.org/se24.

Z. Sirkes and E. Tziperman. Finite difference of adjoint or adjoint of finite difference? *Monthly Weather Review*, 125(12):3373–8, 1997. doi: 10.1175/1520-0493(1997)125⟨3373:FDOAOA⟩2.0.CO;2.

Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 143–152. IEEE, 2006.

Petros Drineas, Michael W Mahoney, Shan Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische mathematik*, 117(2):219–249, 2011.

Michael B Cohen, Jelani Nelson, and David P Woodruff. Optimal approximate matrix product in terms of stable rank. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 55. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

Eric Price, Zhao Song, and David P. Woodruff. Fast regression with an l_∞ guarantee. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 59:1–59:14, 2017. doi: 10.4230/LIPIcs.ICALP.2017.59.

Mert Pilanci and Martin J Wainwright. Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016.

Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-efficient distributed sgd with sketching. *arXiv preprint arXiv:1903.04488*, 2019.