On Learned Sketches for Randomized Numerical Linear Algebra

Simin Liu* Tianrui Liu† Ali Vakilian‡ Yulin Wan§ David P. Woodruff¶

Abstract

We study *"learning-based" sketching* approaches for diverse tasks in numerical linear algebra: least-squares regression, ℓ_p regression, Huber regression, low-rank approximation (LRA), and k-means clustering. Sketching methods are used to quickly and approximately compute properties of large matrices. Linear maps called "sketches" are applied to compress data, and these concise representations are used to compute the desired properties. Specifically, we consider sparse sketches (such as CountSketch).

Recent works have dealt with optimizing sketches for data distributions to perform better than their random counterparts. We extend this theme to several important and ubiquitous tasks, each of which requires a new analysis and novel practical methods. Specifically, our contributions are:

- For all tasks, we introduce fast algorithms using learned sketches with worst-case guarantees. We give a simple task-agnostic method for retaining the worst-case guarantees of randomized sketching, which yields time-optimal algorithms for LRA and least-squares regression. Also, for k-means clustering, we give a faster alternative for retaining worst-case guarantees.
- We show empirically that learned sketches are reliable in improving approximation accuracy, with comparison against "non-learned" sketching baselines.
- We introduce a greedy algorithm for optimizing the location of the nonzero entries of a sparse sketch and prove guarantees for certain distributions on the LRA task. Previous work only looked at optimizing the values rather than the locations. Also, we show empirically that it further improves learned sketch performance.

1 Introduction

Sketching is a powerful approach to dimensionality reduction which guarantees that important properties of the data are preserved. There is much work on designing sketches with approximation guarantees for numerical linear algebra (NLA) tasks including linear regression [Sar06, CW09, CW17, NN13, MM13, Coh16], robust regression [MM13, ACW16], low-rank approximation [WLRT08, HMT11, CW09, CW17, NN13, MM13, CEM+15, Coh16, CW14], and clustering [CEM+15, MMR19]. Also, see the surveys [Mah11, Woo14].

In the basic sketching scheme, we first construct a sketch S, which is typically a random matrix with a very small number of rows. Then, we apply S to an input matrix A by computing SA. Here,

*School of Computer Science, Carnegie Mellon University; siminliu@andrew.cmu.edu.
†Nankai University; 1711435@mail.nankai.edu.cn.
‡Dept. of Computer Science, University of Wisconsin - Madison; vakilian@wisc.edu.
§Anhui University; Y91714026@stu.ahu.edu.cn.
¶School of Computer Science, Carnegie Mellon University; dwoodruf@andrew.cmu.edu.

we note that we only consider sparse sketches, for which SA can be computed in $\mathcal{O}(\text{nnz}(A))$ time, where $\text{nnz}(A)$ denotes the number of non-zero entries of A. Finally, we approximately compute the desired property of A by computing it more quickly on the much smaller SA. The form of S guarantees that SA will be a good proxy for A with respect to the task at hand.

In a recent work, [IVY19] proposed a learning-based method on top of the existing sketching-based framework. While most existing sketches are constructed randomly, [IVY19] showed that learning a sketch for a data distribution can lead to significantly improved performance. They focused on the single task of low rank approximation (LRA), using a time-suboptimal sketching algorithm from [Sar06, CW09]. Using gradient descent, they optimized the values of a sparse sketch.

It remained an open question as to whether such learning-based methods could be applied to a wider range of randomized NLA problems. In this paper, we consider the best-known random sketching algorithms for a wide variety of tasks and design more efficient learned sketches for each of them. In particular, we study learned sketches for least squares regression (single and multiple response), robust regression in the form of Huber and ℓ_p-regression, low rank approximation, and k-means clustering.

Related work on learning based methods. In the last few years, there has been a large body of work on leveraging machine learning to improve the performance of classical algorithms, and we only mention a few here. This includes data-dependent dimensionality reduction, such as approaches for pair-wise/multi-wise similarity preservation for indexing big data [WZSS17] and for general applications [HSYB15]. Machine learning has also been integrated into other related areas such as streaming algorithms [HIKV19, AIV19, JLL$^+$20, CGP20] and compressive sensing [MPB15, BLS$^+$16, BJPD17, MMB17].

2 Preliminaries

Most of the work presented here is based on the following special type of sketching matrix.

Definition 2.1 (CountSketch (CS), sparsity pattern). *The* CountSketch *matrix, $S \in \mathbb{R}^{m\times n}$, is constructed as follows: for each of the n columns, we uniformly randomly pick a row ($p(i) \in [m]$) and a value ($v(i) \in \{-1, 1\}$) for its single non-zero entry. We call $p(i)$ (sometimes denoted $\vec{p} \in \mathbb{Z}_m^n$) the "sparsity pattern" of S, since it represents the positions of its non-zero entries. Likewise, $v(i)$ (sometimes $\vec{v} \in \mathbb{R}^n$) represents the values of the non-zero entries.*

Notation. We write the *singular value decomposition (SVD)* of $A \in \mathbb{R}^{n\times d}$ as $A = U\Sigma V^\top$, with U, V having orthonormal columns, and Σ diagonal with non-negative entries. We write $A^+ = V\Sigma^+U^\top$ to denote A's Moore-Penrose pseudoinverse, where Σ^+ is formed from Σ by reciprocating each of the non-zero diagonal entries, leaving the zeros in place, and transposing. Also, we let $A_k = U_k\Sigma_k V_k^\top$ denote the optimal rank-k approximation of A, where $U_k \in \mathbb{R}^{n\times k}, \Sigma_k \in \mathbb{R}^{k\times k}, V_k \in \mathbb{R}^{d\times k}$ are the truncated versions of U, Σ, V that just include the top k singular vectors/values.

3 Fast Learning-Based Least Squares Regression

We consider a generalized version of the ℓ_2 regression problem known as *multiple-response regression* (MRR). Given a matrix of observations, $A \in \mathbb{R}^{n\times d}$, and a matrix of corresponding values, $B \in$

$\mathbb{R}^{n\times d'}$, the goal of approximate MRR is to compute $\hat{X} \in \mathbb{R}^{d\times d'}$ such that $\left\|A\hat{X} - B\right\|_F^2 \leq (1+\varepsilon)\min_{X\in\mathbb{R}^{d\times d'}} \|AX - B\|_F^2$.

Learning-free sketching algorithm. The learning-free sketching algorithm (1) is simply to sketch A, B with S, a random CS, and compute the closed-form solution on the resulting small matrices.

Algorithm 1 SKETCH-REGRESSION [Sar06, CW17]

Require: $A \in \mathbb{R}^{n\times d}, B \in \mathbb{R}^{n\times d'}, S \in \mathbb{R}^{m\times n}$

1: **return:** $\hat{X} = (SA)^+(SB)$

Lemma 3.1 ([Sar06, CW17]). *Suppose that $S \in \mathbb{R}^{\mathrm{poly}(d/\varepsilon)\times n}$ is a sparse sketching matrix (such as CountSketch) and an affine ε-embedding for A, B. Then,* SKETCH-REGRESSION(A, B, S) *returns a $(1+\varepsilon)$-approximation in time $O(\mathrm{nnz}(A) + \mathrm{nnz}(B) + \mathrm{poly}(\frac{dd'}{\varepsilon}))$.*

Learned sketching algorithm. We optimize our learned sparse sketch (S_L) using gradient descent (Algorithm 2), where $\mathcal{L}(S, A) = \|A(SA)^+(SB) - B\|_F^2$ is the regression objective. Next, we provide a way to retain the worst-case guarantees of random CS while taking advantage of S_L when possible (Algorithm 3). To do so, we efficiently compare the quality of the solution using random CS (S_R) with the one using S_L and choose the better.

Algorithm 2 LEARN-SKETCH: gradient-descent algorithm for optimizing sketch values

Require: $\mathcal{A}_{\mathrm{train}} = \{A_1, ..., A_{N_{\mathrm{train}}}\}$ where $\mathrm{A}_i \in \mathbb{R}^{n\times d}$, learning rate α

1: Initialize $\vec{p}, \vec{v}$ (defined in 2.1) randomly or use Alg. 6 for $\vec{p}$
2: **for** $i = 1$ **to** num_grad_steps **do**
3: **form** S using $\vec{v}, \vec{p}$
4: **sample** batch A_{batch} from A_{train}
5: $\vec{v} \leftarrow \vec{v} - \alpha\frac{\partial\mathcal{L}(S, A_{batch})}{\partial\vec{v}}$
6: **end for**

Algorithm 3 LEARNED-REGRESSION

Require: $A \in \mathbb{R}^{n\times d}, B \in \mathbb{R}^{n\times d'}, S_L \in \mathbb{R}^{\mathrm{poly}(\frac{d}{\varepsilon})\times n}$

1: $\{S_L$ is a learned affine β-embedding matrix of $A, B\}$
2: $S_R \leftarrow \mathrm{CS}(\mathrm{poly}(\frac{d}{\varepsilon}) \times d)$ $\quad$ {CountSketch}
3: $X_L \leftarrow$ SKETCH-REGRESSION(A, B, S_L), $X_R \leftarrow$ SKETCH-REGRESSION(A, B, S_R)
4: $S \leftarrow \mathrm{CS}(\frac{1}{\beta^2} \times n)$, $R \leftarrow \mathrm{CS}(\frac{1}{\beta^2} \times d')$
5: $\Delta_L \leftarrow \left\|S(AX_L - B)R^\top\right\|_F^2$, $\Delta_R \leftarrow \left\|S(AX_R - B)R^\top\right\|_F^2$
6: **if** $\Delta_L \leq \Delta_R$ **then**
7: **return** X_L
8: **end if**
9: **return** X_R

Theorem 3.2 (Learned sketching with guarantees for MRR). *Suppose there exists a learned, sparse, affine matrix S_L computed over $\mathcal{A}_{train}$ with $\text{poly}(\frac{d}{\varepsilon})$ rows that attains a β-approximation over $\mathcal{A}_{\text{test}}$. Then, there exists an algorithm that runs in time $O(\text{nnz}(A)+\text{nnz}(B)+\text{poly}(\frac{dd'}{\varepsilon})+\frac{d}{\beta^4})$ that outputs a $(1+\min(\beta,\varepsilon))$-approximation to the MRR problem.*

Note that when $\beta < \varepsilon$, S_L gives a better approximation error than S_R for the same sketching algorithm runtime. Equivalently, S_L can achieve the same approximation error as S_R with a faster runtime. To see this formally, observe that the non-learned sketching algorithm gives a $(1+\beta)$-approximation in $O(\text{nnz}(A)+\text{nnz}(B)+\text{poly}(\frac{dd'}{\beta}))$ time, versus $O(\text{nnz}(A)+\text{nnz}(B)+\text{poly}(\frac{dd'}{\varepsilon})+\frac{d}{\beta^4})$ time with a learned sketch. Now $\frac{1}{\varepsilon} < \frac{1}{\beta}$, so $\text{poly}(\frac{dd'}{\varepsilon})+\frac{d}{\beta^4} < \text{poly}(\frac{dd'}{\beta})$. In section 8, we show that S_L is reliably better than S_R on natural data sets, so we can solve more quickly than with random CS.

4 Fast Learning-Based Robust Regression

We introduce two extensions of the basic ℓ_2 regression paradigm. Given $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$, the objective is to find a $(1+\varepsilon)$-approximation for $x^* = \text{argmin}_{x\in\mathbb{R}^d} \|Ax-b\|_G$, where G is either a p-norm or the Huber loss, denoted H. Given parameter $\tau > 0$ and an input $y \in \mathbb{R}^n$, the Huber loss is defined as: $\|y\|_H = \sum_{i\in[n]} H(y_i)$, where $H(y_i) = \frac{y_i^2}{2\tau}$ if $|y_i| < \tau$ and $H(y_i) = |y_i| - \frac{\tau}{2}$ otherwise. The Huber loss is popular in robust regression because it combines the advantages of the ℓ_1 and ℓ_2 norms.

Learning-free sketching algorithm. [CW14] shows that sketching with a sparse "M-sketch" gives an $O(1)$-approximation to the optimal ℓ_p and Huber regressors. We describe the construction of an M-sketch: $S \in \mathbb{R}^{O(\text{poly}(k\log n)\times n}$ is formed by vertically stacking $l \in O(\log(n))$ submatrices, $S_0, \ldots, S_l$. $S_0 \in \mathbb{R}^{\text{poly}(k\log n)\times n}$ is a CS matrix and each following S_i retains only $O(1/2^i)$ columns of S_0 and scales them by $O(2^i)$.

Learned sketching algorithm. To train a learned sketch, we initialize a random M-sketch and optimize its non-zero values using gradient descent on the objective function (Algorithm 2). With no closed-form solution, we turn to the Iteratively Reweighted Least Squares (IRLS) algorithm ([HW77]) to compute the optimal regressor. IRLS is a standard heuristic for robust regression that is empirically fast. Its speed is key: the fewer iterations until convergence, the faster it is to calculate an accurate analytic gradient through backpropagation.

5 Fast Learning-Based Low Rank Approximation

Given an input matrix $A \in \mathbb{R}^{n\times d}$ and desired rank k, the goal of approximate LRA is to find a rank-k matrix B such that $\|B-A\|_F^2 \le (1+\varepsilon)\|A-A_k\|_F^2$.

Learning-free sketching algorithm. We consider the time-optimal (up to what are typically considered low order terms) random sketching algorithm by [Sar06, CW17, ACW16] (Algorithm 4).

Algorithm 4 SKETCH-LOWRANK [Sar06, CW17, ACW16].

Require: $A \in \mathbb{R}^{n\times d}, S \in \mathbb{R}^{m_S\times n}, R \in \mathbb{R}^{m_R\times d}$

1: $S_2 \leftarrow \mathrm{CS}(\frac{k^2}{\beta^2}\times n)$, $R_2 \leftarrow \mathrm{CS}(\frac{k^2}{\beta^2}\times d)$ {CountSketch}

2: $U_C \begin{bmatrix} T_C & T_C' \end{bmatrix} \leftarrow S_2AR^\top$, $\begin{bmatrix} T_D^\top \\ T_D'^\top \end{bmatrix} U_D^\top \leftarrow SAR_2^\top$ with U_C, U_D orthogonal

3: $G \leftarrow S_2AR_2^\top$

4: $Z_L'Z_R' \leftarrow [U_C^\top G U_D]_k$

5: $Z_L = \begin{bmatrix} Z_L' T_D^{-\top} & 0 \end{bmatrix}, Z_R = \begin{bmatrix} T_C^{-1} Z_R' \\ 0 \end{bmatrix}$

6: $Z = Z_L Z_R$

7: **return:** $AR^\top ZSA$ in form $P_{n\times k}, Q_{k\times d}$

Lemma 5.1. *Given CS matrices $S \in \mathbb{R}^{\mathrm{poly}(k/\varepsilon)\times d}$ and $R \in \mathbb{R}^{\mathrm{poly}(k/\varepsilon)\times d}$, Algorithm 4 runs in $O(\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(k/\varepsilon))$ time and with constant probability returns a $(1+\varepsilon)$-approximate* rank-k *approximation of A.*

Learned sketching algorithm. As before, we optimize our sketches (S, R, S_2, R_2) using gradient descent (Algorithm 2) and use a comparison method to retain the worst-case guarantees of random sketching (similar to Algorithm 3. Our algorithm is the first learning-based algorithm to achieve an optimal running time; the previous algorithm of [IVY19] does not.

Theorem 5.2 (Low-Rank Approximation). *Suppose that there exists a learned, sparse, affine matrix S_L computed over $\mathcal{A}_{train}$ with $\mathrm{poly}(\frac{k}{\varepsilon})$ rows which attains a $(1+\beta)$-approximation over A_{test}. Then, there is an algorithm $\mathcal{A}$ that runs in time $O(\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(\frac{k}{\varepsilon}) + \frac{k^4}{\beta^4}\cdot\mathrm{poly}(\frac{k}{\varepsilon}))$ that outputs a $(1+\min(\beta,\varepsilon))$-approximate rank-$k$ approximation of A.*

6 Fast Learning-Based k-means Clustering

Let $A \in \mathbb{R}^{n\times d}$ represent a set of n points, $A_1, \ldots, A_n \in \mathbb{R}^d$. In approximate k-means clustering, the goal is to find a partition of $A_1, ..., A_n$ into k clusters $\hat{\mathcal{C}} = \{\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_k\}$ such that $\mathrm{cost}(\hat{\mathcal{C}}) \le (1+\varepsilon)\mathrm{cost}(\mathcal{C}^*) := \min_C \sum_{i=1}^k \min_{\mu_i\in\mathbb{R}^d} \sum_{j\in C_i} \|A_j - \mu_i\|_2^2$, where μ_i denotes the center of cluster $\mathcal{C}_i$. Specifically, each cluster C_i consists of the indices of the points assigned to it.

Learning-free sketching algorithm. Our algorithm is simply to right-sketch A using CS $R \in \mathbb{R}^{O(k^2/\varepsilon^2)\times d}$, compressing the n points to a smaller dimension. Then, we use any existing approximation algorithm for k-means. In our experiments, we chose k-means++ with Lloyd's algorithm ([AV07], [Llo82]).
Cohen et al. [CEM+15] showed that the cost of any clustering on $AR^\top$ is a $(1+\varepsilon)$-approximation of its cost on A; hence, the approximation guarantee of any k-means algorithm (including k-means++) increases by a factor of $(1+\varepsilon)$.

Learned sketching algorithm. To train a learned sketch, we use Algorithm 2 with $\mathcal{L}(S, A)$ as the LRA objective. Suppose that $R \in \mathbb{R}^{m\times d}$, with $m \in \mathrm{poly}(k/\varepsilon)$, is optimized so that there is a good rank-m approximation to A in $col(AR^\top)$. This suggests that $col(AR^\top)$ and $col(U_m)$ (where

$A = U\Sigma V^\top$) are similar subspaces. Cohen et al. [CEM+15] showed that sketching by transforming A to an approximate top singular vector space yields a good solution to k-means.
Next, we give an algorithm for approximate k-means using the learned sketch while retaining the worst-case guarantees of the random sketch (5). The idea is to concatenate a random sketch (R_1) to our learned sketch (R_2).

Algorithm 5 LEARNED-SKETCH-k-MEANS

Require: $A \in \mathbb{R}^{n\times d}, \mathcal{A}_{\text{train}} = \{A_1, ..., A_{N_{\text{train}}}\}, k \in \mathbb{Z}^+$
1: $R_1 \leftarrow$ LEARN-SKETCH($\mathcal{A}_{train}$)
2: $R_2 \leftarrow CS(\text{poly}(\frac{k}{\varepsilon} \times d)$ {CountSketch}
3: $U\Sigma V^T \leftarrow A \begin{bmatrix} R_1 & R_2 \end{bmatrix}$
4: **return:** k-MEANS++(AV, k)

In the following, we show that this concatenated sketch performs at least as well as the random sketch alone. In other words, the sketching matrices for k-means clustering satisfy a so-called "sketch monotonicity" property introduced in [IVY19].
Next, we use the following notation. Given a matrix U with orthogonal columns, let $\pi_U(A) = AUU^\top$, which is the projection of the rows of A onto $col(U)$. Let C_U be the optimal k-means clustering of $\pi_U(A)$.

Theorem 6.1 (Sketch monotonicity property for k-means). *Assume we have $A \in \mathbb{R}^{n\times d}$. We also have random CountSketch $S \in \mathbb{R}^{O(\text{poly}(k/\varepsilon))\times n}$ and define $U \in \mathbb{R}^{d\times O(\text{poly}(k/\varepsilon))}$ with orthogonal columns such that $col(U) = row(SA)$. Then, any extension of S to S' (for example, concatenation with a learned CountSketch S_L) yields a better approximate k-means approximation. Specifically, define W with orthogonal columns such that $col(W) = row(S'A)$. Let C^* denote the optimal partition of A, C_U denote the optimal partition of $\pi_U(A)$, and C_W denote the optimal partition of $\pi_W(A)$. Then*

$$cost(C_W) \leq (1+\varepsilon)cost(C_U) \leq (1+O(\varepsilon))cost(C^*)$$

7 Greedy initialization

Previous work on learned sparse sketches only optimized the *values* of a sketch's non-zero entries and not their *placement*, which was randomized ([IVY19]). Empirically, we find that it is better to optimize both the locations (a sketch's "sparsity pattern") and the values (Tables 8.1, 8.6). We also give examples of data distributions where this greedy initialization is guaranteed to be better than random for the LRA task.
Algorithm. We apply Algorithm 6 to optimize the sparsity pattern before optimizing values (Algorithm 2), during which the sparsity pattern is fixed. Applying a CS matrix ($SA, S \in \mathbb{R}^{m\times n}$) can be viewed as weighting each row of A and hashing it to one of m bins, where the rows are summed. The greedy algorithm is simple: we iterate over row indices in some order and for each, we evaluate the task objective ($\mathcal{L}(S, A)$) for all possible *(weight, bin)* assignments. Then, we add a new entry to the sketch for the best option. In Algorithm 6, $\mathcal{D}_w$ is the set of candidate weights, which we set to 10 samples in $[-2, 2]$. An advantage of this method is that it is task-agnostic, requiring only the ability to evaluate the task objective function.

Algorithm 6 GREEDY-INIT

Require: $\mathcal{A}_{\text{train}} = \{A_1, ..., A_{N_{\text{train}}}\}$ where $A_i \in \mathbb{R}^{n \times d}$

initialize $\vec{p} = \mathbf{0}_n, \vec{v} = \mathbf{0}_n$

sample A_{batch} from $\mathcal{A}_{train}$

for $i = 1$ **to** n **do**

  form S using $\vec{p}, \vec{v}$

  $w^*, j^* = \text{argmin}_{w \in \mathcal{D}_w, j \in [m]} \sum_{A_i \in A_{batch}} \mathcal{L}(S_{w,j}, A_i)$ where $S_{w,j} = S + w(\vec{e_j}\vec{e_i}^\top)$

  $\vec{v}[i] \leftarrow w^*, \vec{p}[i] \leftarrow j^*$

end for

Guarantees on natural distributions. We present two natural distributions and show that for both, greedy-initialized CS produces a better approximate low-rank approximation than CS with a random sparsity pattern.

1. Spiked Covariance Model. In this model which is is introduced by Johnstone [Joh01], each matrix has covariance $\text{diag}(\ell_1, \cdots, \ell_r, 1, \cdots, 1)$ with r small. That is, each matrix has a few eigenvalues that are significantly larger than the rest. This distribution closely models certain types of data in speech recognition [HBT95, Joh01], wireless communication [Tel99, CH12], mathematical finance [PGR$^+$02, LCPB00, MS04] and statistical learning [HR04a, HR04b, BS06, Pau07, WF17].

We consider $A \in \mathbb{R}^{n \times d}$ from a spiked covariance distribution $\mathcal{A}_{sp}(s, \ell)$, where $s = O(k)$ is the number of "heavy" rows with norm ℓ and the remaining rows are of norm 1. Also, we assume the direction of each row is picked uniformly at random.

Theorem 7.1. *Consider a matrix $A \in \mathcal{A}_{sp}(s, \ell)$. Let S_g denote a CS constructed using Algorithm 6 that visits the rows of A in order of non-increasing row norm. Let S_r denote a random CS. Then, there exists a fixed constant η such that* $\min_{\text{rank-}k\, X \in \text{rowsp}(S_g A)} \|X - A\|_F^2 \leq (1 - \eta) \min_{\text{rank-}k\, X \in \text{rowsp}(S_r A)} \|X - A\|_F^2$.

2. Heavy Tailed (Zipfian) Random Matrices. This model consists of matrices with Zipfian-distributed squared row norms. Real data also often follows this model [AG08, BP09, AGZ10, Ver10, BJ11, AT16].

We consider symmetric matrices $A \in \mathbb{R}^{n \times n}$ with "Zipfian squared" row norms where $\forall i \in O(\log n)$, there are 2×2^i rows of squared norm $n^2/4^i$. We assume the direction of each row is picked uniformly at random.

Theorem 7.2. *Consider a symmetric (random) matrix $A \in \mathbb{R}^{n \times n}$ with Zipfian squared norms. Let S_g denote the CS constructed using Algorithm 6 that visits the rows of A in order of non-increasing row norm. Let S_r denote a random CS. Then, there exists a fixed constant η such that* $\min_{\text{rank-}k\, X \in \text{rowsp}(S_g A)} \|X - A\|_F^2 \leq (1 - \eta) \min_{\text{rank-}k\, X \in \text{rowsp}(S_r A)} \|X - A\|_F^2$.

Remark 7.3. Though our analysis shows that the greedy algorithm with *non-increasing row norm ordering* gives a better sketch than the standard approach in certain cases, in other cases, the converse may be true. See the appendix for an example where greedy with random ordering yields a cost that is a constant factor better than the non-increasing row norm ordering.

8 Experiments

See the appendix for additional experimental details (Section D). Our code is available publicly.[1]

8.1 Datasets

We use assorted real-world data sets to verify the broad effectiveness of our algorithms.

For the regression family of tasks (MMR, Huber, ℓ_p):

- **Gas:** time series of measured and simulated greenhouse gas concentrations in California's atmosphere. Each (A, B) corresponds to a different measurement location. A's columns are consecutive measurements and B's are consecutive simulated values. $|(A,B)_{\text{train}}| = 400, |(A,B)_{\text{test}}| = 100, A \in \mathbb{R}^{327\times 15}$, $B \in \mathbb{R}^{327\times 1}$.[2]
- **Tunnel:** time series of gas concentrations measured by eight sensors in a wind tunnel. Each (A, B) corresponds to a different data collection trial. A's columns are consecutive measurements of temperature, relative humidity, and time, as well as values from two sensors; B's columns are values from the other six sensors. $|(A,B)_{\text{train}}| = 144, |(A,B)_{\text{test}}| = 36, A \in \mathbb{R}^{13530\times 5}$, $B \in \mathbb{R}^{13530\times 6}$.[3]
- **Electric:** residential electric load measurements. Each (A, B) corresponds to a different residence. Matrix columns are consecutive measurements from different days. $|(A,B)_{\text{train}}| = 250, |(A,B)_{\text{test}}| = 63, A \in \mathbb{R}^{950\times 50}$, $B \in \mathbb{R}^{950\times 50}$.[4]

For the low-rank approximation family of tasks (LRA, k-means):

- **Videos - Logo, Friends, Eagle**: frames from YouTube videos of a logo being painted, a scene from the TV show *Friends*, and the feed from an eagle's nest camera. Each A is formed by flattening a $1920 \times 1080 \times 3$ RGB array. $A \in \mathbb{R}^{5760\times 1080}, |A_{\text{train}}| = 400, |A_{\text{test}}| = 100$.[5]
- **Hyper:** hyperspectral images depicting outdoor scenes. $A \in \mathbb{R}^{1024\times 768}, |A_{\text{train}}| = 400, |A_{\text{test}}| = 100$.[6]

Some matrices in these aforementioned data sets have much larger singular values than others. To keep the approximation errors at the same magnitude, we normalized the matrices to have the same largest singular value.

For the section on greedy sensitivity analysis:

- **Synthetic spiked covariance**: given LRA parameter k, each $A \in \mathbb{R}^{n\times d}$ has random vectors with norm $O(\sqrt{\frac{n}{k}})$ for its first $O(k)$ rows. The rest of the rows are random unit vectors. $A \in \mathbb{R}^{290\times 5000}, |A_{\text{train}}| = 160, |A_{\text{train}}| = 40$.

[1] `https://github.com/sliu2019/learned_sketch`

[2] `https://archive.ics.uci.edu/ml/datasets/Greenhouse+Gas+Observing+Network`

[3] `https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+exposed+to+turbulent+gas+mixtures`

[4] `https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014`

[5] `http://youtu.be/L5HQoFIaT4I`, `http://youtu.be/xmLZsEfXEgE`, `http://youtu.be/ufnf_q_3Ofg6`

[6] `https://github.com/gistairc/HS-SOD`

8.2 Baselines

We describe the sketches used in the algorithms we compare:

- **Random**: random CS
- **Learned (random pattern)**: a sparse sketch with learned values, random positions for non-zero entries
- **Learned (greedy pattern)**: a sparse sketch where both values and positions of non-zero entries are learned.

We also consider additional baselines with non-oblivious (matrix-dependent) sketches for LRA and k-means clustering:

- **Exact SVD**: Randomly sample $A_i \in A_{\text{train}}$, compute $A_i = U\Sigma V^T$ and sketch as AV_m.
- **Column sampling**: sketch as AR, where R is computed from randomly sampled $A_i \in A_{\text{train}}$. R selects columns of A_i with probability proportional to squared column norm and scales by its inverse.

8.3 Results

The following tables contain average values and standard deviations computed over 3 trials. Experimental parameters (i.e., learning rate for gradient descent) can be found in the appendix.

Regression family of tasks. For MRR (Table 8.1), *learned (random pattern)* gives a $20-50\%$ improvement over *random*. Also, *learned (greedy pattern)* is either comparable to *learned (random pattern)* or attains up to 10% improvement. For ℓ_p ($p = 1.5$) and Huber regression, we observe that the learned sketch almost always yields better performance than the random one. For $\ell_{1.5}$ (Table 8.2), there is up to 15% improvement; for Huber (Table 8.3), there is $30-95\%$ improvement.

Table 8.1: Test errors for MRR

m, Sketch	Gas	Tunnel	Electric
10, random	106.038	0.236	14.942
10, learned (random pattern)	**78.357**	0.177	11.041
10, learned (greedy pattern)	87.2379	**0.1559**	**10.842**
20, random	44.180	0.0949	10.663
20, learned (random pattern)	**30.3709**	**0.0580**	8.822
20, learned (greedy pattern)	36.7858	0.0596	**7.9203**
30, random	18.2751	0.0476	9.173
30, learned (random pattern)	**12.5393**	0.0368	7.497
30, learned (greedy pattern)	14.3316	**0.03200**	**6.8032**

LRA family of tasks. For the time-optimal approximate LRA algorithm (Algorithm 4), *learned (random pattern)* gives a $35-75\%$ improvement over *random* (Table 8.6). The work [IVY19] learns sketches for a different approximation algorithm involving just one sketch (their Algorithm

Table 8.2: Test errors for $\ell_{1.5}$ regression

m, Sketch	Gas	Tunnel	Electric
20, random	16.100757	0.905101	139.079274
20, learned (random pattern)	**15.662183**	**0.767999**	**131.755758**
30, random	14.955649	0.639754	57.653981
30, learned (random pattern)	**14.858939**	**0.505537**	**48.685349**
40, random	14.329586	0.425036	48.733799
40, learned (random pattern)	**14.305626**	**0.345803**	**44.492352**

Table 8.3: Test errors for Huber regression

m, Sketch	Gas	Tunnel	Electric
20, random	0.058685	0.001115	0.288651
20, learned (random pattern)	**0.042292**	**0.001091**	**0.275271**
30, random	0.045003	0.001279	0.108427
30, learned (random pattern)	**0.020599**	**0.000987**	**0.106119**
40, random	0.043164	0.001006	0.073449
40, learned (random pattern)	**0.009170**	**0.000918**	**0.071874**

1), as opposed to four. Though the use of more sketches increases approximation error (compare Table 8.4 to Table 8.6), it is much faster (see Table 8.5). Now, the desired trade-off between speed and accuracy varies depending on the application, so we leave it to the reader to decide which algorithm is more advantageous.

In Table 8.6, we compare our approach, *learned (greedy pattern)*, with several baselines on LRA. Our approach is always better or comparable to *learned (random pattern)* from [IVY19], showing improvements of $10 - 40\%$. It also beats all baselines, except on the **Eagle** data set, where *exact SVD* is best. However, our approach has better time complexity than *exact SVD*, since our sketch is sparse and can be applied in input sparsity time (see Table D.5).

For approximate k-means clustering (Table 8.7), we note that *learned (random pattern)* is always better than *random*. It is also always the best or a close second to *exact SVD*. However, as we mentioned above, the solution using *exact SVD* has the disadvantage of being slower to compute (see Table D.5).

Table 8.4: Test errors for LRA (using Algorithm 4 with four sketches)

k, m, Sketch	Logo	Eagle	Friends	Hyper
$20, 20$, random	6.389263 ± 0.117776	12.782090 ± 0.472347	11.442603 ± 0.641905	14.133427 ± 0.601569
$20, 20$, learned (random pattern)	$\mathbf{1.874400} \pm 0.018637$	$\mathbf{3.126370} \pm 0.030783$	$\mathbf{3.026187} \pm 0.036555$	$\mathbf{8.059653} \pm 0.063467$
$20, 40$, random	2.329610 ± 0.064810	8.437103 ± 0.195867	4.100357 ± 0.172226	6.939890 ± 0.211195
$20, 40$, learned (random pattern)	$\mathbf{0.918453} \pm 0.011491$	$\mathbf{2.084160} \pm 0.048539$	$\mathbf{1.392333} \pm 0.019683$	$\mathbf{4.455760} \pm 0.036090$
$30, 30$, random	4.713050 ± 0.099571	12.535033 ± 0.074421	8.794927 ± 0.173146	12.328263 ± 0.079653
$30, 30$, learned (random pattern)	$\mathbf{2.119253} \pm 0.031725$	$\mathbf{4.286367} \pm 0.037845$	$\mathbf{3.574900} \pm 0.034765$	$\mathbf{8.252257} \pm 0.008888$
$30, 60$, random	1.649640 ± 0.020364	7.921303 ± 0.104303	2.752303 ± 0.041925	5.855720 ± 0.058504
$30, 60$, learned (random pattern)	$\mathbf{0.916483} \pm 0.006059$	$\mathbf{2.636563} \pm 0.045209$	$\mathbf{1.397930} \pm 0.006837$	$\mathbf{4.240370} \pm 0.012680$

Sensitivity analysis of the greedy algorithm. We explore how sensitive the performance of *learned (greedy pattern)* is to row ordering. We consider five orderings: random, by decreasing row norm, by decreasing row leverage score, backwards (largest row indices first), and forwards. We

Table 8.5: Timing comparison for two approximate LRA algorithms

n, d	1-sketch time (sec)	4-sketch time (sec)
$5000, 4000$	0.18944 ± 0.11338	$\mathbf{0.03369} \pm 0.00436$
$2500, 2000$	0.07076 ± 0.07540	$\mathbf{0.02097} \pm 0.00445$
$1250, 1000$	0.04645 ± 0.03839	$\mathbf{0.01437} \pm 0.00401$

Table 8.6: Test errors for LRA (using Algorithm 1 from [IVY19] with one sketch)

k, m, Sketch	Logo	Eagle	Friends	Hyper
$20, 20,$ random	3.251	5.97	5.615	6.949
$20, 20,$ exact SVD	1.331	**0.3027**	1.7139	29.1169
$20, 20,$ learned (random pattern)	0.712	0.468	1.928	2.91
$20, 20,$ learned (greedy pattern)	**0.7121**	0.3977	**1.5233**	**2.6812**
$20, 40,$ random	0.930	3.36	1.542	2.971
$20, 40,$ exact SVD	0.5580	**0.1458**	0.5658	23.7912
$20, 40,$ learned (random pattern)	0.255	0.344	0.723	1.273
$20, 40,$ learned (greedy pattern)	**0.1961**	0.2287	**0.4066**	**0.7841**
$30, 30,$ random	2.366	5.84	4.476	6.115
$30, 30,$ exact SVD	1.2435	**0.3290**	1.8100	27.1428
$30, 30,$ learned (random pattern)	0.857	0.762	2.212	3.116
$30, 30,$ learned (greedy pattern)	**0.7552**	0.5326	**1.6814**	**2.3989**
$30, 60,$ random	0.650	3.08	1.0575	2.315
$30, 60,$ exact SVD	0.4925	**0.1402**	0.5086	23.7702
$30, 60,$ learned (random pattern)	0.290	0.485	0.713	1.274
$30, 60,$ learned (greedy pattern)	**0.1978**	0.3340	**0.4064**	**0.7174**

Table 8.7: Test errors for k-means clustering

k, m, Sketch	Logo	Eagle	Friends	Hyper
$3, 20,$ random	0.591470 ± 0.003440	1.070480 ± 0.003670	0.975360 ± 0.005860	1.070570 ± 0.003100
$3, 20,$ column sampling	0.602550 ± 0.013000	1.096580 ± 0.007790	1.034400 ± 0.050050	1.079490 ± 0.001800
$3, 20,$ exact SVD	$\mathbf{0.581520} \pm 0.000210$	1.052440 ± 0.000680	0.979120 ± 0.001280	1.058340 ± 0.003350
$3, 20,$ learned (random pattern)	0.583310 ± 0.000600	$\mathbf{1.048180} \pm 0.000080$	$\mathbf{0.975170} \pm 0.006990$	$\mathbf{1.058340} \pm 0.000630$
$6, 40,$ random	0.453670 ± 0.000440	0.998330 ± 0.002930	0.785100 ± 0.004010	0.864000 ± 0.004150
$6, 40,$ column sampling	0.473200 ± 0.004230	1.010780 ± 0.015490	0.801710 ± 0.004370	0.881960 ± 0.005430
$6, 40,$ exact SVD	$\mathbf{0.451430} \pm 0.000600$	0.991750 ± 0.000210	0.787060 ± 0.000580	$\mathbf{0.854410} \pm 0.002290$
$6, 40,$ learned (random pattern)	0.452130 ± 0.000970	$\mathbf{0.984460} \pm 0.000930$	$\mathbf{0.777650} \pm 0.000540$	0.855280 ± 0.001150
$10, 70,$ random	0.353370 ± 0.000720	0.934050 ± 0.002240	0.635600 ± 0.000910	0.733080 ± 0.001470
$10, 70,$ column sampling	0.375130 ± 0.010740	0.939450 ± 0.002650	0.667460 ± 0.018620	0.757350 ± 0.025370
$10, 70,$ exact SVD	0.351280 ± 0.000480	0.932540 ± 0.000890	0.637190 ± 0.002200	$\mathbf{0.727370} \pm 0.001240$
$10, 70,$ learned (random pattern)	$\mathbf{0.351130} \pm 0.000290$	$\mathbf{0.927560} \pm 0.000610$	$\mathbf{0.633310} \pm 0.000310$	0.728620 ± 0.000420

find that on some data sets (**Logo**), the random ordering is best, while on others (**Friends**), the decreasing row norm ordering is better. We also show that for the spiked covariance distribution of section 7, the empirical results corroborate the theory, which states that decreasing row norm is better than random.

9 Conclusions

In this work, we developed methods and principles for learning sketches on a variety of numerical linear algebra tasks, including least-squares, ℓ_p, and Huber regression, low-rank approximation,

Table 8.8: Test errors for *learned (greedy pattern)* with different row orders

Row ordering	Logo	Friends	Synthetic spiked covariance
random	**0.712100** $\pm$ 0.013720	1.523310 $\pm$ 0.057720	2.548000 $\pm$ 0.696460
decreasing row norm	0.764400 $\pm$ 0.042230	**1.372240** $\pm$ 0.087160	0.589950 $\pm$ 0.441570
decreasing leverage score	0.726540 $\pm$ 0.019410	1.539450 $\pm$ 0.147030	1.409430 $\pm$ 0.147600
backwards	0.838640 $\pm$ 0.038600	1.543980 $\pm$ 0.177190	2.039050 $\pm$ 0.410570
forwards	0.939110 $\pm$ 0.098140	1.748210 $\pm$ 0.148670	**0.199500** $\pm$ 0.025460

and k-means clustering. For least-squares regression and low rank approximation, the proposed algorithms are time-optimal. For all tasks, we provided a simple means to retain the worst-case guarantees of classical sketching algorithms. We also showed that a learned sketch is reliably (and sometimes significantly) better than classical random sketches on several types of data sets. Further, we provided a method for optimizing the nonzero entry placement in our sparse sketches and showed for low rank approximation that this empirically and provably yields additional gains. Possible directions for future research include devising alternate algorithms for optimizing sketches over distributions and also extending these methods to other linear algebra problems where sketching has proven useful (e.g., regularized regression, kernel regression, and tensor decomposition).

Acknowledgments

The authors would like to thank Piotr Indyk and Samson Zhou for insightful discussions about the project. The authors would also like to thank the anonymous reviewers for their valuable suggestions on an earlier version. D. P. Woodruff was supported in part by Office of Naval Research (ONR) grant N00014-18-1-256.

References

[ACW16] Haim Avron, Kenneth L Clarkson, and David P Woodruff. Sharper bounds for regularized data fitting. *arXiv preprint arXiv:1611.03225*, 2016.

[AG08] Gérard Ben Arous and Alice Guionnet. The spectrum of heavy tailed random matrices. *Communications in Mathematical Physics*, 278(3):715–751, 2008.

[AGZ10] Greg W Anderson, Alice Guionnet, and Ofer Zeitouni. *An introduction to random matrices*, volume 118. Cambridge university press, 2010.

[AIV19] Anders Aamand, Piotr Indyk, and Ali Vakilian. (learned) frequency estimation algorithms under zipfian distribution. *arXiv preprint arXiv:1908.05198*, 2019.

[AT16] Antonio Auffinger and Si Tang. Extreme eigenvalues of sparse, heavy tailed random matrices. *Stochastic Processes and their Applications*, 126(11):3310–3330, 2016.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the 18th Annual ACM SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007.

[BJ11] Zdzislaw Burda and Jerzy Jurkiewicz. Heavy-tailed random matrices. In *The Oxford Handbook of Random Matrix Theory*. Oxford University Press, 2011.

[BJPD17] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546, 2017.

[BLS+16] Luca Baldassarre, Yen-Huan Li, Jonathan Scarlett, Baran Gözcü, Ilija Bogunovic, and Volkan Cevher. Learning-based compressive subsampling. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):809–822, 2016.

[BP09] Jean-Philippe Bouchaud and Marc Potters. Financial applications of random matrix theory: a short review. *arXiv preprint arXiv:0910.1205*, 2009.

[Bra06] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415.1, 2006.

[BS06] Jinho Baik and Jack W Silverstein. Eigenvalues of large sample covariance matrices of spiked population models. *Journal of multivariate analysis*, 97(6):1382–1408, 2006.

[CEM+15] Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172, 2015.

[CGP20] Edith Cohen, Ofir Geri, and Rasmus Pagh. Composable sketches for functions of frequencies: Beyond the worst case. *arXiv preprint arXiv:2004.04772*, 2020.

[CH12] Romain Couillet and Walid Hachem. Fluctuations of spiked random matrix models and failure diagnosis in sensor networks. *IEEE Transactions on Information Theory*, 59(1):509–525, 2012.

[Coh16] Michael B. Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 278–287, 2016.

[CW09] Kenneth L Clarkson and David P Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual symposium on Theory of computing (STOC)*, pages 205–214, 2009.

[CW14] Kenneth L. Clarkson and David P. Woodruff. Sketching for m-estimators: A unified approach to robust regression. In *Proceedings of the 26th Annual ACM SIAM Symposium on Discrete Algorithms*, pages 921–939, 2014.

[CW17] Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):54, 2017.

[FSS13] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the 24th annual ACM-SIAM symposium on Discrete Algorithms*, pages 1434–1453, 2013.

[HBT95] Trevor Hastie, Andreas Buja, and Robert Tibshirani. Penalized discriminant analysis. *The Annals of Statistics*, pages 73–102, 1995.

[HIKV19] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. *International Conference on Learning Representations*, 2019.

[HMT11] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[HR04a] David Hoyle and Magnus Rattray. Limiting form of the sample covariance eigenspectrum in pca and kernel pca. In *Advances in Neural Information Processing Systems*, pages 1181–1188, 2004.

[HR04b] David C Hoyle and Magnus Rattray. Principal-component-analysis eigenvalue spectra from data with symmetry-breaking structure. *Physical Review E*, 69(2):026124, 2004.

[HSYB15] Chinmay Hegde, Aswin C. Sankaranarayanan, Wotao Yin, and Richard G. Baraniuk. Numax: A convex approach for learning near-isometric linear embeddings. In *IEEE Transactions on Signal Processing*, pages 6109–6121, 2015.

[HW77] Paul W. Holland and Roy E. Welsch. Robust regression using iteratively reweighted least-squares. In *Communications in Statistics - Theory and Methods*, pages 813–827, 1977.

[IVY19] Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-based low-rank approximations. In *Advances in Neural Information Processing Systems*, pages 7400–7410, 2019.

[JLL+20] Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P. Woodruff. Learning-augmented data stream algorithms. In *International Conference on Learning Representations*, 2020.

[Joh01] Iain M Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *Annals of statistics*, pages 295–327, 2001.

[LCPB00] Laurent Laloux, Pierre Cizeau, Marc Potters, and Jean-Philippe Bouchaud. Random matrix theory and financial correlations. *International Journal of Theoretical and Applied Finance*, 3(03):391–397, 2000.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[Mah11] Michael W Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[MM13] Xiangrui Meng and Michael W Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 91–100, 2013.

[MMB17] Chris Metzler, Ali Mousavi, and Richard Baraniuk. Learned d-amp: Principled neural network based compressive image recovery. In *Advances in Neural Information Processing Systems*, pages 1772–1783, 2017.

[MMR19] Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1027–1038, 2019.

[MPB15] Ali Mousavi, Ankit B Patel, and Richard G Baraniuk. A deep learning approach to structured signal recovery. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 1336–1343. IEEE, 2015.

[MS04] Yannick Malevergne and Didier Sornette. Collective origin of the coexistence of apparent random matrix theory noise and of factors in large sample correlation matrices. *Physica A: Statistical Mechanics and its Applications*, 331(3-4):660–668, 2004.

[NN13] Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 117–126, 2013.

[Pau07] Debashis Paul. Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistica Sinica*, pages 1617–1642, 2007.

[PGR+02] Vasiliki Plerou, Parameswaran Gopikrishnan, Bernd Rosenow, Luis A Nunes Amaral, Thomas Guhr, and H Eugene Stanley. Random matrix approach to cross correlations in financial data. *Physical Review E*, 65(6):066126, 2002.

[Sar06] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 143–152, 2006.

[Tel99] Emre Telatar. Capacity of multi-antenna gaussian channels. *European transactions on telecommunications*, 10(6):585–595, 1999.

[Ver10] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[WF17] Weichen Wang and Jianqing Fan. Asymptotics of empirical eigenstructure for high dimensional spiked covariance. *Annals of statistics*, 45(3):1342, 2017.

[WLRT08] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366, 2008.

[Woo14] David P Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[WZSS17] Jingdong Wang, Ting Zhang, Nicu Sebe, and Heng Tao ShenWang. A survey on learning to hash. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 769–790, 2017.

A Missing Proofs

A.1 Missing Proofs in Section 3

Definition A.1 (Affine Embedding). *Given a pair of matrices A and B each with n rows, a matrix S is an* affine ε-embedding *if for all X of appropriate shape, $\|S(AX-B)\|_F^2 = (1\pm\varepsilon)\|AX-B\|_F^2$.*

The following result is shown in [CW17] and sharpened with [NN13, MM13].

Lemma A.2. *Given matrices A, B with n rows, a sparse embedding matrix (i.e., CS) with $O(\mathrm{rank}(A)^2/\varepsilon^2)$ rows is an affine ε-embedding matrix with constant probability. Moreover, the matrix product $S\cdot A$ can be computed in $O(\mathrm{nnz}(A))$ time, where $\mathrm{nnz}(A)$ denotes the number of non-zero entries of matrix A. We also define two sets of matrices, A_{train} and A_{test}. Only A_{train} is made available at the time of sketch optimization.*

Lemma A.3 ([CW17]; Lemma 40). *Let A be an $n\times d$ matrix and let $S\in\mathbb{R}^{O(\frac{1}{\varepsilon^2})\times n}$ be a randomly chosen sparse embedding matrix (i.e., CS). Then with constant probability, $\|SA\|_F^2 = (1\pm\varepsilon)\|A\|_F^2$.*

Proof of **Lemma 3.1.** Since S is an affine ε-embedding matrix of A, B, then

$$\|SAX-SB\|_F^2 = (1\pm\varepsilon)\|AX-B\|_F^2$$

Next, by the normal equations, $(SA)^+(SB)$ is a minimizer of $\min_X\|SAX-SB\|_F^2$.

To bound the runtime, note that since S is a sparse sketching matrix (i.e., CountSketch) we can compute SA and SB in time $O(\mathrm{nnz}(A)+\mathrm{nnz}(B))$ and reduce the problem to an instance of multiple-response regression with m rows. Then, we can solve the reduced size problem in time $O(d\cdot d'\cdot m^2)$: $O(d\cdot m^2)$ to compute $(SA)^+$ and $O(d\cdot d'\cdot m^2)$ to compute $(SA)^+(SB)$. □

Proof of **Theorem 3.2.** By Lemma A.2, a CountSketch S_O with $\mathrm{poly}(\frac{d}{\varepsilon})$ rows is an affine ε-embedding matrix of A, B with constant probability.

Next, let X_L and X_O be respectively the solutions returned by SKETCH-REGRESSION(A, B, S_L) and SKETCH-REGRESSION(A, B, S_O). By Lemma 3.1, with constant probability, the following hold:

$$\min(\|AX_L-B\|_F^2, \|AX_O-B\|_F^2) \le \begin{cases} (1+\beta)\min_X\|AX-B\|_F^2 & \text{if } (A,B)\in\mathcal{D} \\ (1+\varepsilon)\min_X\|AX-B\|_F^2 & \text{otherwise} \end{cases} \tag{A.1}$$

Hence, it only remains to compute the minimum of $\|AX_L-B\|_F^2$ and $\|AX_O-B\|_F^2$ efficiently. Note that it takes $\Omega(n\cdot d\cdot d')$ to compute these values. However, for our purpose it suffices to compare $(1+\beta)$-estimates of these values and return the minimum of the estimates. To achieve this, we use two applications of Lemma A.3 with $R^\top\in\mathbb{R}^{O(\frac{1}{\beta^2})\times d'}, S\in\mathbb{R}^{O(\frac{1}{\beta^2})\times n}$. For any X' (in particular, both X_O and X_L), with constant probability,

$$\left\|S(AX'-B)R\right\|_F^2 = \left\|R^\top(AX'-B)^\top S^\top\right\|_F^2 = (1\pm\beta)\left\|(AX'-B)\right\|_F^2 \tag{A.2}$$

Let Δ_L and Δ_O respectively denote $\|S(AX_L - B)R\|_F^2$ and $\|S(AX_O - B)R\|_F^2$. By Eq. (A.2) and union bound over of X_O and X_L, with constant probability,

$$\begin{aligned}\min(\Delta_L, \Delta_O) &\leq (1 + O(\beta)) \cdot \min(\|AX_L - B\|_F^2, \|AX_O - B\|_F^2) \\ &\leq \begin{cases} (1 + O(\beta)) \min_X \|AX - B\|_F^2 & \text{if } (A, B) \in \mathcal{D} \\ (1 + O(\varepsilon)) \min_X \|AX - B\|_F^2 & \text{otherwise} \end{cases}\end{aligned}$$

The last inequality holds by Eq. (A.1) and $\beta < \varepsilon$.

Runtime Analysis. By Lemma 3.1, X_O an X_L can be computed in $\text{nnz}(A) + \text{nnz}(B) + \text{poly}(\frac{dd'}{\varepsilon})$ time. Next, the time to compute Δ_L and Δ_O is

$$O(\text{nnz}(A) + \text{nnz}(XR) + \text{nnz}(B) + \frac{d}{\beta^4}) = O(\text{nnz}(A) + \text{nnz}(B) + dd' + \frac{d}{\beta^4}),$$

where X is either X_O or X_L and we use that fact that $\text{nnz}(X) \leq d \cdot d'$ (i.e., the total number of cells in X).

Thus, the total runtime of Algorithm 3 is $O(\text{nnz}(A) + \text{nnz}(B) + \text{poly}(\frac{dd'}{\varepsilon}) + \frac{d}{\beta^4})$. □

Corollary A.4 (Regression). *Suppose that there exists a learned sparse affine β-approximation matrix S_L with $\text{poly}(\frac{d}{\varepsilon})$ rows for the matrices in family $\mathcal{D}$ where $\beta < \varepsilon$. Then, there exists an algorithm $\mathcal{A}$ that runs in time $O(\text{nnz}(A) + \text{nnz}(B) + \text{poly}(\frac{d}{\varepsilon}) + \frac{d}{\beta^2})$ such that,*

1. **Improved bound:** *if $(A, b) \in \mathcal{D}$, then $\mathcal{A}$ outputs a $(1 + \beta)$-approximation of the regression problem.*
2. **Worst-case bound:** *otherwise, $\mathcal{A}$ outputs a $(1+\varepsilon)$-approximation of the regression problem.*

Proof: The proof is similar to the proof of Theorem 3.2 but here the sketching matrix S suffices. This improves the dependence in the runtime on β to β^{-2}. Moreover, here $d' = 1$. □

A.2 Missing Proofs in Section 5

Lemma A.5. *Suppose that $S \in \mathbb{R}^{m_S \times n}$ and $R \in \mathbb{R}^{m_R \times d}$ are sparse affine ε-embedding matrices for $(A^\top, A)$ and $((SA)^\top, A^\top)$. Then,*

$$\min_{\text{rank-}k\, X} \left\| AR^\top XSA - A \right\|_F^2 \leq (1 + \varepsilon) \|A_k - A\|_F^2$$

Proof: Consider the following multiple-response regression problem:

$$\min_{\text{rank-}k\, X} \|A_k X - A\|_F^2 . \tag{A.3}$$

Note that since $X = I_k$ is a feasible solution to Eq. (A.3), $\min_{\text{rank-}k\, X} \|A_k X - A\|_F^2 = \|A_k - A\|_F^2$. Let $S \in \mathbb{R}^{m_S \times n}$ be a sketching matrix that satisfies the condition of Lemma A.7 for $A := A_k$ and $B := A$. By the normal equations, the rank-k minimizer of $\|SA_k X - SA\|_F^2$ is $(SA_k)^+ SA$. Hence,

$$\left\| A_k (SA_k)^+ SA - A \right\|_F^2 \leq (1 + \varepsilon) \|A_k - A\|_F^2 , \tag{A.4}$$

which in particular shows that a $(1+\varepsilon)$ rank-k approximation of A exists in the row space of SA. In other words,

$$\min_{\text{rank-}k\, X} \|XSA - A\|_F^2 \leq (1+\varepsilon)\, \|A_k - A\|_F^2 . \tag{A.5}$$

Next, let $R \in \mathbb{R}^{m_R \times d}$ be a sketching matrix that satisfies the condition of Lemma A.7 for $A := (SA)^\top$ and $B := A^\top$. Let Y denote the rank-k minimizer of $\left\|R(SA)^\top X^\top - RA^\top\right\|_F^2$. Hence,

$$\begin{aligned}
\left\|(SA)^\top Y^\top - A^\top\right\|_F^2 &\leq (1+\varepsilon) \min_{\text{rank-}k\, X} \|XSA - A\|_F^2 && \triangleright \text{Lemma A.7} \\
&\leq (1 + O(\varepsilon))\, \|A_k - A\|_F^2 && \triangleright \text{Eq. (A.5)}
\end{aligned} \tag{A.6}$$

Note that by the normal equations, again $\operatorname{rowsp}(Y^\top) \subseteq \operatorname{rowsp}(RA^\top)$ and we can write $Y = AR^\top Z$ where $\operatorname{rank}(Z) = k$. Thus,

$$\begin{aligned}
\min_{\text{rank-}k\, X} \left\|AR^\top XSA - A\right\|_F^2 &\leq \left\|AR^\top ZSA - A\right\|_F^2 \\
&= \left\|(SA)^\top Y^\top - A^\top\right\|_F^2 && \triangleright\, Y = AR^\top Z \\
&\leq (1 + O(\varepsilon))\, \|A_k - A\|_F^2 && \triangleright \text{Eq. (A.6)}
\end{aligned}$$ □

Lemma A.6 ([ACW16]; Lemma 27). *For $C \in \mathbb{R}^{p \times m'}, D \in \mathbb{R}^{m \times p'}, G \in \mathbb{R}^{p \times p'}$, the following problem*

$$\min_{\text{rank-}k\, Z} \|CZD - G\|_F^2 \tag{A.7}$$

can be solved in $O(pm'r_C + p'mr_D + pp'(r_D + r_C))$ time, where $r_C = \operatorname{rank}(C) \leq \min\{m', p\}$ and $r_D = \operatorname{rank}(D) \leq \min\{m, p'\}$.

Proof: Let U_C and $U_D^\top$ be orthogonal bases for $\operatorname{colsp}(C)$ and $\operatorname{rowsp}(D)$, respectively, so that for each Z, $CZD = U_C Z' U_D^\top$ for some Z'. Let P_C and P_D be the projection matrices onto the subspaces spanned by the rows of $C^\top$ and $D^\top$, respectively: $P_C = U_C U_C^\top$ and $P_D = U_D U_D^\top$. Then by the Pythagorean theorem,

$$\begin{aligned}
\|CZD - G\|_F^2 &= \left\|P_C U_C Z' U_D^\top P_D - G\right\|_F^2 \\
&= \left\|P_C U_C Z' U_D^\top P_D - P_C G P_D\right\|_F^2 + \|P_C G(I - P_D)\|_F^2 + \|(I - P_C)G\|_F^2 ,
\end{aligned}$$

where the first equality holds since $P_C U_C = U_C$ and $U_D^\top P_D = U_D^\top$ and the second equality follows from the Pythagorean theorem. Hence,

$$\operatorname{argmin}_{\text{rank-}k\, Z} \|CZD - G\|_F^2 = \operatorname{argmin}_{\text{rank-}k\, Z} \left\|P_C U_C Z U_D^\top P_D - P_C G P_D\right\|_F^2 .$$

Moreover,

$$\left\|P_C U_C Z U_D^\top P_D - P_C G P_D\right\|_F^2 = \left\|U_C Z U_D^\top - U_C U_C^\top G U_D U_D^\top\right\|_F^2 = \left\|Z - U_C^\top G U_D\right\|_F^2 ,$$

where the first equality holds since $U_C^\top U_C = I$ and $U_D^\top U_D = I$, and the second equality holds since U_C and $U_D^\top$ are orthonormal. Hence,

$$\operatorname{argmin}_{\text{rank-}k\, Z} \|CZD - G\|_F^2 = \operatorname{argmin}_{\text{rank-}k\, Z} \left\| Z - U_C^\top G U_D \right\|_F^2 .$$

Next, we can find $Z = [U_C^\top G U_D]_k$ by computing the SVD of $U_C^\top G U_D$.

Runtime analysis. We can compute U_C and U_D by the Gram-Schmidt process in time $O(pm' r_C + p' m r_D)$, and $U_C^\top G U_D$ in time $O(\min\{r_C p'(p + r_D), r_D p(p' + r_C)\})$. Finally, the time to compute Z (i.e., an SVD computation of $U_C^\top G U_D$) is $O(r_C r_D \cdot \min\{r_C, r_D\})$. Since $r_C \le \min\{p, m'\}$ and $r_D \le \min\{p', m\}$, the total runtime to minimize Z in Eq. (A.7) is $O(pm' r_C + p' m r_D + pp'(r_C + r_D))$.□

Proof of **Lemma 5.1** It first computes $C = S_2 A R^\top, D = S A R_2^\top, G = S_2 A R_2^\top$ which can be done in time $O(\operatorname{nnz}(A))$. As an example, we bound the time to compute $C = S_2 A R$. Note that since S_2 is a sparse sketching matrix (i.e., CountSketch), $S_2 A$ can be computed in $O(\operatorname{nnz}(A))$ time and the number of non-zero entries in the resulting matrix is at most $\operatorname{nnz}(A)$. Hence, since R is a sparse sketching matrix as well, C can be computed in time $O(\operatorname{nnz}(A) + \operatorname{nnz}(S_2 A)) = O(\operatorname{nnz}(A))$. Then, it takes an extra $\operatorname{poly}(k/\varepsilon) \cdot \frac{k^2}{\beta^2}$ time to store C, D and G in matrix form. Next, as we showed in Lemma A.6, the time to compute Z in Algorithm 4 is $O(\frac{k^4}{\beta^4} \cdot \operatorname{poly}(k/\varepsilon))$. Finally, it takes $(n + d)\operatorname{poly}(k/\varepsilon)$ time to return the solution in the form of $P_{n\times k} Q_{k\times d}$. Hence, the total runtime is $O(\operatorname{nnz}(A) + (n + d)\operatorname{poly}(k/\varepsilon) + \frac{k^4}{\beta^4} \cdot \operatorname{poly}(k/\varepsilon))$. The exact analysis goes through for the approximation guarantee and we omit it here. □

Lemma A.7 ([ACW16]; Lemma 25). *Suppose that $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times d'}$. Moreover, let S be an oblivious sparse affine ε-embedding matrix (i.e., a CountSketch matrix) with $(\operatorname{rank}(A)^2/\varepsilon^2)$ rows. Then with constant probability,*

$$\tilde{X} = \operatorname{argmin}_{\text{rank-}k\, X} \|SAX - SB\|_F^2 ,$$

satisfies

$$\left\| A\tilde{X} - B \right\|_F^2 \le (1 + \varepsilon) \min_{\text{rank-}k\, X} \|AX - B\|_F^2 .$$

In other words, in $O(\operatorname{nnz}(A) + \operatorname{nnz}(B)) + (d + d')(\operatorname{rank}(A)^2/\varepsilon^2)$ time, we can reduce the problem to a smaller (multi-response regression) problem with $(\operatorname{rank}(A)^2/\varepsilon^2)$ rows whose solution is a $(1 + \varepsilon)$-approximate solution to the original problem.

Proof of **Theorem 5.2.** Let S_O and R_O be CountSketch matrices of size $\operatorname{poly}(k/\varepsilon) \times n$ and $\operatorname{poly}(k/\varepsilon) \times d$. Note that since $\operatorname{rank}(A_k) = k$ and $\operatorname{rank}((S_O A)^\top) \le \operatorname{poly}(k/\varepsilon)$, S_O and R_O are respectively affine ε-embedding matrices of (A_k, A) and $((S_O A)^\top, A^\top)$. Then, by an application of Lemma A.5

$$\min_{\text{rank-}k\, X} \left\| A R_O^\top X S_O A - A \right\|_F^2 \le (1 + O(\varepsilon)) \|A_k - A\|_F^2 \tag{A.8}$$

Similarly, by the properties mentioned for S_L and R_L and an application of Lemma A.5 if $(A_k, A) \in \mathcal{D}$ and $((S_L A)^\top, A^\top) \in \mathcal{D}'$, then

$$\min_{\text{rank-}k\, X} \left\| A R_L^\top X S_L A - A \right\|_F^2 \le (1 + O(\beta)) \|A_k - A\|_F^2 \tag{A.9}$$

Next, by applications of Lemma A.2, we multiply Eq. (A.9) from both sides by oblivious *sparse* affine β-embedding matrices (i.e., CountSketch) S_2, R_2 with $O(k^2/\beta^2)$ rows,

$$\min_{\text{rank-}k\, X} \left\| S_2(AR_L^\top X S_L A)R_2^\top - S_2 A R_2^\top \right\|_F^2 = (1 \pm O(\beta)) \min_{\text{rank-}k\, X} \left\| S_2 A R_L^\top X S_L A - S_2 A \right\|_F^2 \tag{A.10}$$

where the first inequality follows since R_2 is an affine β-embedding for $(SA)^\top$ which is of rank $\text{poly}(k/\varepsilon)$ and the second inequality follows since S_2 is an affine β-embedding for $AR^\top$ which is of rank $\text{poly}(k/\varepsilon)$. By a similar argument,

$$\min_{\text{rank-}k\, X} \left\| S_2(AR_O^\top X S_O A)R_2^\top - S_2 A R_2^\top \right\|_F^2 = (1 + O(\varepsilon)) \min_{\text{rank-}k\, X} \|A_k - A\|_F^2 , \tag{A.11}$$

Next, let (P_L, Q_L) and (P_O, Q_O) be respectively the rank-k approximations of A in factored form using (S_L, R_L) and (S_O, R_O) (see Algorithm 4). Then, Eq. (A.10) together with Eq. (A.11) implies that

$$\min(\|P_L Q_L - A\|_F^2 , \|P_O Q_O - A\|_F^2) = \begin{cases} (1 \pm O(\beta)) \|A_k - A\|_F^2 & \text{if } (A_k, A) \in \mathcal{D}, \\ & \text{and } (A^\top S^\top, A^\top) \in \mathcal{D}' \\ (1 \pm O(\varepsilon)) \|A_k - A\|_F^2 & \text{otherwise} \end{cases} \tag{A.12}$$

Hence, it only remains to compute the minimum of $\|P_L Q_L - A\|_F^2$ and $\|P_O Q_O - A\|_F^2$ efficiently and we proceed similarly to the proof of Theorem 3.2. We use two applications of Lemma A.3 with $R^\top \in \mathbb{R}^{O(\frac{1}{\beta^2}) \times d}, S \in \mathbb{R}^{O(\frac{1}{\beta^2}) \times n}$: $\Delta_L := \|S(P_L Q_L - A)R\|_F^2$ and $\Delta_O := \|S(P_O Q_O - A)R\|_F^2$. Hence,

$$\begin{aligned} \min(\Delta_L, \Delta_O) &\le (1 + O(\beta)) \cdot \min(\|P_L Q_L - A\|_F^2 , \|P_O Q_O - A\|_F^2) \\ &\le \begin{cases} (1 + O(\beta)) \|A_k - A\|_F^2 & \text{if } (A_k, A) \in \mathcal{D}, \\ & \text{and } (A^\top S^\top, A^\top) \in \mathcal{D}' \\ (1 + O(\varepsilon)) \|A_k - A\|_F^2 & \text{otherwise} \end{cases} \end{aligned}$$

The last inequality follows from Eq. (A.12) and $\beta < \varepsilon$.

Runtime analysis. By Lemma 5.1, Algorithm 4 computes P_L, Q_L and P_O, Q_O in $O(\text{nnz}(A) + (n+d)\,\text{poly}(\frac{k}{\varepsilon}) + (\frac{k^4}{\beta^4}) \cdot \text{poly}(\frac{k}{\varepsilon}))$.

Next, it takes $O(\text{nnz}(A) + (n+d) \cdot k + \frac{k}{\beta^4})$ to compute Δ_L and Δ_O. As an example, we bound the amount of time required to compute $SP_L Q_L R - SAR$ corresponding to Δ_L. Since S and R are sparse sketching matrices, SP_L, $Q_L R$ and SAR can be computed in $\text{nnz}(SP_L)+\text{nnz}(Q_L R)+\text{nnz}(A)$. Since SP_L and $Q_L R$ are respectively of size $\frac{1}{\beta^2} \times k$ and $k \times \frac{1}{\beta^2}$, in total it takes $O(\text{nnz}(A) + \frac{k}{\beta^2})$ to compute these three matrices. Then, we can compute $SP_L Q_L R$ and $\|SP_L Q_L R SAR\|_F$ in time $O(\frac{k}{\beta^4})$.

Hence, the total runtime of Algorithm 4 is $O(\text{nnz}(A) + (n+d) \cdot \text{poly}(\frac{k}{\varepsilon}) + (\frac{k^4}{\beta^4}) \cdot \text{poly}(\frac{k}{\varepsilon}))$. □

A.3 Missing Proofs in Section 6

We restate notation and the main result below for ease of reference.

Notation. We define A_m as the optimal m-rank approximation of A formed by truncated SVD: $A_m = U_m \Sigma_m V_m^\top$.
Given a matrix U with orthogonal columns, let $\pi_U(A) = AUU^\top$, which is the projection of the rows of A onto $col(U)$. Let C_U be the optimal k-means partition of $\pi_U(A)$. Further, we let $\mu_{C_{U,i}}$ denote the i-th cluster's center in the optimal k-means clustering on $\pi_U(A)$.
We denote $\text{dist}^2(A, \mu)$ as the k-means loss given cluster centers (and their corresponding partition):

$$\text{dist}^2(A, \mu) = \sum_{i \in [k]} \sum_{j \in C_i} \|A_j - \mu_i\|_2^2$$

Likewise, $\text{cost}(C)$ is the k-means loss given a partition:

$$\text{cost}(C) = \sum_{i \in [k]} \min_{\mu_i} \sum_{j \in C_i} \|A_j - \mu_i\|_2^2$$

Definition A.8 (Projection-cost preserving sketch). *$\tilde{A}$ is a projection-cost preserving sketch of A if for any low rank projection matrix P and c not dependent on P:*

$$(1-\varepsilon)\|A - PA\|_F^2 \leq \left\|\tilde{A} - P\tilde{A}\right\|_F^2 + c \leq (1+\varepsilon)\|A - PA\|_F^2$$

Theorem (6.1: Sketch monotonicity property for k-means). *Assume we have $A \in \mathbb{R}^{n \times d}$. We also have random CountSketch $S \in \mathbb{R}^{O(\text{poly}(k/\varepsilon)) \times n}$ and define $U \in \mathbb{R}^{d \times O(\text{poly}(k/\varepsilon))}$ with orthogonal columns such that $\text{colsp}(U) = \text{rowsp}(SA)$. Then, any extension of S to S' (for example, concatenation with a learned CountSketch S_L) yields a better approximate k-means approximation. Specifically, define W with orthogonal columns such that $col(W) = row(S'A)$. Let C^* denote the optimal partition of A, C_U denote the optimal partition of $\pi_U(A)$, and C_W denote the optimal partition of $\pi_W(A)$. Then*

$$cost(C_W) \leq (1 + O(\varepsilon))cost(C_U) \leq (1 + O(\varepsilon))cost(C^*)$$

Proof:

$$\begin{aligned}
\text{cost}(C_W) &= \sum_{i \in [k]} \min_{\mu_i} \sum_{j \in C_{W,i}} \|A_j - \mu_i\|^2 \\
&\leq \sum_{i \in [k]} \sum_{j \in C_{W,i}} \left\|A_j - \mu_{C_{W,i}}\right\|^2 \\
&= \sum_{i \in [k]} \sum_{j \in C_{W,i}} \|A_j - \pi_W(A_j)\|^2 + \left\|\pi_W(A_j) - \mu_{C_{W,i}}\right\|^2 \qquad \text{(A.13)} \\
&\leq \sum_{i \in [k]} \sum_{j \in C_{U,i}} \|A_j - \pi_W(A_j)\|^2 + \left\|\pi_W(A_j) - \mu_{C_{U,i}}\right\|^2 \qquad \text{(A.14)} \\
&= \sum_{i \in [k]} \sum_{j \in C_{U,i}} \left\|A_j - \mu_{C_{U,i}}\right\|^2 \qquad \text{(A.15)} \\
&\leq (1+\varepsilon) \sum_{i \in k} \min_{\mu_i} \sum_{j \in C_{U,i}} \|A_j - \mu_i\|^2 \qquad \text{(A.16)} \\
&= (1+\varepsilon)\text{cost}(C_U) \\
&\leq (1 + O(\varepsilon))\text{cost}(C^*)
\end{aligned}$$

□

(A.13): $\mu_{C_W} \in \text{colsp}(W)$ so we can apply the Pythagorean Theorem.
(A.14): (μ_{C_W}, C_W) is an optimal k-means clustering of the projected points $\pi_W(A)$.
(A.15): $\mu_{C_U} \in \text{colsp}(U) \subset \text{colsp}(W)$, so we can apply the Pythagorean Theorem.
(A.16): We apply Corollary A.12.

Remark A.9. Our result shows that the "sketch monotonicity" property holds for sketching matrices that provide strong coresets for k-means clustering. Besides strong coresets, an alternate approach to showing that the clustering objective is approximately preserved on sketched inputs is to show a weaker property: the clustering cost is preserved for *all possible partitions* of the points into k groups [MMR19]. While the dimension reduction mappings satisfying strong coresets require $\text{poly}(k/\varepsilon)$ dimensions, [MMR19] shows that $O(\log k/\varepsilon^2)$ dimensions suffice to satisfy this "partition" guarantee. An interesting question for further research is if the sketch monotonicity guarantee also applies to the construction of [MMR19].

Corollary A.10. *Assume we have $A \in \mathbb{R}^{n\times d}$, $j \in \mathbb{Z}^+$, $\varepsilon > 0$. Define $m = \min(O(\text{poly}(j/\varepsilon)), d)$. Let $\tilde{A}_m$ be a $(1+\varepsilon)$-approximation to A_m of the form $\tilde{A}_m = AVV^\top$ where $SA = U\Sigma V^\top$ for CountSketch $S \in \mathbb{R}^{m\times n}$. Let $X \in \mathbb{R}^{d\times j}$ be a matrix whose columns are orthonormal, and let $Y \in \mathbb{R}^{d\times(d-j)}$ be a matrix with orthonormal columns that spans the orthogonal complement of colsp(X). Then*

$$\left\|AXX^\top - \tilde{A}_m XX^\top\right\|_F^2 \leq \varepsilon \cdot \|AY\|_F^2.$$

Proof:

$$\begin{aligned}
\left\|AXX^\top - \tilde{A}_m XX^\top\right\|_F^2 &= \left\|A(I-VV^\top)XX^\top\right\|_F^2 \\
&\leq j\left\|A(I-VV^\top)XX^\top\right\|_2^2 && \text{(A.17)}\\
&\leq j\left\|A(I-VV^\top)\right\|_2^2 && \text{(A.18)}\\
&\leq j\cdot O(\frac{\varepsilon}{j})\|A-A_j\|_F^2 && \text{(A.19)}\\
&= O(\varepsilon)\sum_{i=j}^{\min(n,d)}\sigma_i^2 && \text{(A.20)}\\
&\leq O(\varepsilon)\|AY\|_F^2 && \text{(A.21)}
\end{aligned}$$

□

(A.17) Note that $\text{rank}(X) = j$, so $\text{rank}(A(I-VV^\top)XX^\top) = j$. We use this fact to bound the Frobenius norm by the operator norm.
(A.18) Using the fact that $XX^\top$ is a projection.
(A.19) Using Lemma 18 from [CEM+15], where A_j is the optimal rank-j approximation to A. We can apply this lemma because CountSketch is one of the eligible types of random projection matrices.
(A.20) Letting σ_i be the singular values of A.
(A.21) $\sum_{i=j}^{\min(n,d)}\sigma_i^2 = \min_Y \|AY\|_F^2$ for $Y \in \mathbb{R}^{d\times(d-j)}$ with orthonormal columns.

Theorem A.11. *Assume we have $A \in \mathbb{R}^{n\times d}$, $j \in \mathbb{Z}^+$, $\varepsilon \in (0,1]$. Define $m = \min(O(\text{poly}(j/\varepsilon)), d)$. Let $\tilde{A}_m$ be a $(1+\varepsilon)$-approximation to A_m of the form $\tilde{A}_m = AVV^\top$ where $SA = U\Sigma V^\top$ for CountSketch $S \in \mathbb{R}^{m\times n}$. Then, for any non-empty set μ contained in a j-dimensional subspace, we have:*

$$\left|\text{dist}^2(\tilde{A}_m, \mu) + \left\|\tilde{A}_m - A\right\|_F^2 - \text{dist}^2(A, \mu)\right| \leq \varepsilon \,\text{dist}^2(A, \mu)$$

Proof: We follow the proof of Theorem 22 in [FSS13], but substitute different analyses in place of Corollaries 16 and 20. The result of [FSS13] involves the *best* rank-m approximation of A, A_m; we will show it for the *approximate* rank-m approximation, $\tilde{A}_m$.
Define $X \in \mathbb{R}^{d\times j}$ with orthonormal columns such that $\text{colsp}(X) = \text{span}(\mu)$. Likewise, define $Y \in \mathbb{R}^{d\times(d-j)}$ with orthonormal columns such that $\text{colsp}(Y) = \text{span}(\mu)^\perp$. By the Pythagorean theorem:

$$\text{dist}^2(\tilde{A}_m, \mu) = \|\tilde{A}_m Y\|_F^2 + \text{dist}^2(\tilde{A}_m XX^T, \mu)$$

and

$$\text{dist}^2(A, \mu) = \|AY\|_F^2 + \text{dist}^2(AXX^T, \mu). \tag{A.22}$$

Hence,

$$\begin{aligned}
&\left|\left(\text{dist}^2(\tilde{A}_m, \mu) + \left\|A - \tilde{A}_m\right\|_F^2\right) - \text{dist}^2(A, \mu)\right| \\
=&\left|\|\tilde{A}_m Y\|_F^2 + \text{dist}^2(\tilde{A}_m XX^T, \mu) + \left\|A - \tilde{A}_m\right\|_F^2 - \left(\|AY\|_F^2 + \text{dist}^2(AXX^T, \mu)\right)\right| \\
\leq&\left|\|\tilde{A}_m Y\|_F^2 + \|A - \tilde{A}_m\|_F^2 - \|AY\|_F^2\right| + \left|\text{dist}^2(\tilde{A}_m XX^T, \mu) - \text{dist}^2(AXX^T, \mu)\right| && \text{(A.23)} \\
\leq&\frac{\varepsilon^2}{8} \cdot \|AY\|_F^2 + \left|\text{dist}^2(\tilde{A}_m XX^T, \mu) - \text{dist}^2(AXX^T, \mu)\right| && \text{(A.24)} \\
\leq&\frac{\varepsilon^2}{8} \cdot \text{dist}^2(A, \mu) + \left|\text{dist}^2(\tilde{A}_m XX^T, \mu) - \text{dist}^2(AXX^T, \mu)\right| \quad \triangleright \text{ Used (A.22)} && \text{(A.25)}
\end{aligned}$$

(A.23) Triangle inequality.
(A.24) Take ϵ in Theorem 16 from [CEM+15] as $\varepsilon^2/8$. This theorem implies that $\tilde{A}_m$ is a projection-cost preserving sketch with the c term as $\left\|A - \tilde{A}_m\right\|_F^2$. Specifically, it says AV is a project-cost preserving sketch, which means $\tilde{A}_m = AVV^\top$ is too: V has orthonormal columns so $\|(I-P)AV\|_F^2 = \left\|(I-P)AVV^\top\right\|_F^2$.

By Corollary A.10,

$$\left\|\tilde{A}_m XX^T - AXX^T\right\|_F^2 \leq \frac{\varepsilon^2}{8} \cdot \|AY\|_F^2.$$

Since $\mu \in \text{rowsp}(X)$, we have $\|AY\|_F^2 \leq \text{dist}^2(A, \mu)$. Using Corollary 21 from [FSS13] while taking ε as $\varepsilon/4$, A as $\tilde{A}_m XX^T$, and B as AXX^T yields

$$|\,\text{dist}^2(\tilde{A}_m XX^T, \mu) - \text{dist}^2(AXX^T, \mu)| \leq \frac{\varepsilon}{4} \cdot \text{dist}^2(AXX^T, \mu) + (1 + \frac{4}{\varepsilon}) \cdot \left\|\tilde{A}_m XX^T - AXX^T\right\|_F^2$$

By A.22, $\text{dist}^2(AXX^T, \mu) \leq \text{dist}^2(A, \mu)$. Finally, we combining the last two inequalities with (A.25):

$$
\begin{aligned}
&\left|\left(\operatorname{dist}^2(\tilde{A}_m,\mu)+\left\|A-\tilde{A}_m\right\|_F^2\right)-\operatorname{dist}^2(A,\mu)\right| \\
&\leq \frac{\varepsilon^2}{8}\cdot\operatorname{dist}^2(A,\mu)+\frac{\varepsilon}{4}\cdot\operatorname{dist}^2(A,\mu)+\frac{\varepsilon^2}{8}\cdot(1+\frac{4}{\varepsilon})\cdot\operatorname{dist}^2(A,\mu) \\
&\leq \varepsilon\cdot\operatorname{dist}^2(A,\mu),
\end{aligned}
$$

where in the last inequality we used the assumption $\varepsilon \leq 1$. □

Corollary A.12. *Assume we have $A \in \mathbb{R}^{n\times d}$ and CountSketch $S \in \mathbb{R}^{O(\mathrm{poly}(k/\varepsilon))\times n}$. Then, define $U \in \mathbb{R}^{d\times O(\mathrm{poly}(k/\varepsilon))}$ with orthogonal columns spanning row(SA). Also define $A^U = \pi_U(A)$ and μ_U as the set of optimal cluster centers found on A^U. Now, assume $\varepsilon \in (0, 1/3]$. Then, μ_U is a $(1+\varepsilon)$-approximation to the optimal k-means clustering of A. That is, defining μ_U^* as the cluster centers which minimize the cost of partition C_U on A, we have:*

$$
\operatorname{dist}^2(A,\mu_U) \leq (1+\varepsilon)\operatorname{dist}^2(A,\mu_U^*)
$$

Proof: By using $\frac{\varepsilon}{3}$ in Theorem A.11 with j as k,

$$
\left|\operatorname{dist}^2(A^U,\mu_U)+\left\|A-A^U\right\|_F^2-\operatorname{dist}^2(A,\mu_U)\right| \leq \frac{\varepsilon}{3}\operatorname{dist}^2(A,\mu_U)
$$

which implies that

$$
(1-\frac{\varepsilon}{3})\operatorname{dist}^2(A,\mu_U) \leq \operatorname{dist}^2(A^U,\mu_U)+\left\|A-A^U\right\|_F^2 \tag{A.26}
$$

Likewise, by Theorem A.11 on A^U and μ_U^* (and taking j as k),

$$
\left|\operatorname{dist}^2(A^U,\mu_U^*)+\left\|A-A^U\right\|_F^2-\operatorname{dist}^2(A,\mu_U^*)\right| \leq \frac{\varepsilon}{3}\operatorname{dist}^2(A,\mu_U^*)
$$

which implies that

$$
\operatorname{dist}^2(A^U,\mu_U^*)+\left\|A-A^U\right\|_F^2 \leq (1+\frac{\varepsilon}{3})\operatorname{dist}^2(A,\mu_U^*) \tag{A.27}
$$

By (A.26) and (A.27) together, we have:

$$
\begin{aligned}
(1-\frac{\varepsilon}{3})\operatorname{dist}^2(A,\mu_U) &\leq \operatorname{dist}^2(A^U,\mu_U)+\left\|A-A^U\right\|_F^2 \\
&\leq \operatorname{dist}^2(A^U,\mu_U^*)+\left\|A-A^U\right\|_F^2 \\
&\leq (1+\frac{\varepsilon}{3})\operatorname{dist}^2(A,\mu_U^*)
\end{aligned}
$$

Now, $\frac{1+\varepsilon/3}{1-\varepsilon/3} \leq 1+\varepsilon$, so we have $\operatorname{dist}^2(A,\mu_U) \leq (1+\varepsilon)\operatorname{dist}^2(A,\mu_U^*)$. □

B Spectral Norm Guarantee for Zipfian Matrices

In this section, we show that if the singular values of the input matrix A follow a Zipfian distribution (i.e., $\sigma_i \propto i^{-\alpha}$ for a constant α), we can find a $(1+\varepsilon)$ rank-k approximation of A with respect to the *spectral* norm. A key theorem in this section is the following.

Theorem B.1 ([CEM+15], Theorem 27). *Given an input matrix $A \in \mathbb{R}^{n\times d}$, there exists an algorithm that runs in $O(\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(k/\varepsilon))$ and returns a projection matrix $P = QQ^\top$ such that with constant probability the following holds:*

$$\|AP - A\|_2^2 \le (1+\varepsilon)\,\|A - A_k\|_2^2 + O(\frac{\varepsilon}{k})\,\|A - A_k\|_F^2 \tag{B.1}$$

Next, we prove the main claim in this section. There are various ways to prove this, using, for example, a bound on the *s*table rank of A; we give the following short proof for completeness.

Theorem B.2. *Given a matrix $A \in \mathbb{R}^{n\times d}$ whose singular values follow a Zipfian distribution (i.e., $\sigma_i \propto i^{-\alpha}$) with a constant $\alpha \ge 1/2$, there exists an algorithm that computes a* rank-k *matrix B (in factored form) such that $\|A - B\|_2^2 \le (1+\varepsilon)\,\|A - A_k\|_2^2$.*

Proof: Note that since the singular values of A follow a Zipfian distribution with parameter α, for any value of k,

$$\begin{aligned}
\|A - A_k\|_F^2 = \sum_{i=k+1}^{\mathrm{rank}(A)} \sigma_i^2 &= C \cdot \sum_{i=k+1}^{\mathrm{rank}(A)} i^{-2\alpha} && \triangleright\ \sigma_i = \sqrt{C}/i^\alpha \\
&\le C \cdot \int_k^{\mathrm{rank}(A)} x^{-2\alpha}\,dx \\
&\le k \cdot \frac{1}{2\alpha - 1} \cdot (1 + \frac{1}{k})^{2\alpha} \cdot C/(k+1)^{2\alpha} \\
&= O(k \cdot \sigma_{k+1}^2) \\
&= O(k \cdot \|A - A_k\|_2^2)
\end{aligned} \tag{B.2}$$

By an application of Theorem B.1, we can compute a matrix B in a factored form in time $O(\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(k/\varepsilon))$ such that with constant probability,

$$\begin{aligned}
\|B - A\|_2^2 &\le (1+\varepsilon)\,\|A - A_k\|_2^2 + O(\frac{\varepsilon}{k})\,\|A - A_k\|_F^2 && \triangleright\ \text{By Eq. (B.1)} \\
&\le (1 + O(\varepsilon))\,\|A - A_k\|_2^2 && \triangleright\ \text{Eq. (B.2)}
\end{aligned}$$

□

C Greedy Initialization

In this section, we analyze the performance of the greedy algorithm on the two distributions described in Section 7.

Preliminaries and Notation. Left-multiplying A by CountSketch $S \in \mathbb{R}^{m\times n}$ is equivalent to hashing the rows of A to m bins with coefficients in $\{-1, 1\}$. The greedy algorithm proceeds through the rows of A (in some order) and decides which bin to hash to, denoting this by adding an entry to S. We will denote the bins as b_i and their summed contents as w_i.

C.1 Spiked covariance model with sparse left singular vectors.

To recap, every matrix $A \in \mathbb{R}^{n\times d}$ from the distribution $\mathcal{A}_{sp}(s,\ell)$ has $s < k$ "heavy" rows $(A_{r_1}, \cdots, A_{r_s})$ of norm $\ell > 1$. The indices of the heavy rows can be arbitrary, but must be the same for all members of the distribution and are unknown to the algorithm. The remaining rows (called "light" rows) have unit norm.

In other words: let $\mathcal{R} = \{r_1, \ldots, r_s\}$. For all rows $A_i, i \in [n]$:

$$A_i = \begin{cases} \ell \cdot v_i & \text{if } i \in \mathcal{R} \\ v_i & \text{o.w.} \end{cases}$$

where v_i is a uniformly random unit vector.

We also assume that $S_r, S_g \in \mathbb{R}^{k\times n}$ and non-increasing row norm ordering for the greedy algorithm.

Proof sketch. First, we show that the greedy algorithm using a non-increasing row norm ordering will isolate heavy rows (i.e., each is alone in a bin). Then, we conclude by showing that this yields a better k-rank approximation error when d is sufficiently large compared to n. We begin with some preliminary observations that will be of use later.

It is well known that a set of uniformly random vectors are *ε-almost orthogonal* (i.e., the magnitudes of their pairwise inner products are at most ε).

Observation C.1. *Let $v_1, \cdots, v_n \in \mathbb{R}^d$ be a set of random unit vectors. Then with high probability $|\langle v_i, v_j\rangle| \le 2\sqrt{\frac{\log n}{d}}, \forall\, i < j \le n$.*

We define $\overline{\varepsilon} = 2\sqrt{\frac{\log n}{d}}$.

Observation C.2. *Let $u_1, \cdots, u_t$ be a set of vectors such that for each pair of $i < j \le t$, $|\langle \frac{u_i}{\|u_i\|}, \frac{u_j}{\|u_j\|}\rangle| \le \varepsilon$, and $g_i, \cdots, g_j \in \{-1, 1\}$. Then,*

$$\sum_{i=1}^{t} \|u_i\|_2^2 - 2\varepsilon \sum_{i<j\le t} \|u_i\|_2 \|u_j\|_2 \le \left\| \sum_{i=1}^{t} g_i u_i \right\|_2^2 \le \sum_{i=1}^{t} \|u_i\|_2^2 + 2\varepsilon \sum_{i<j\le t} \|u_i\|_2 \|u_j\|_2 \tag{C.1}$$

Next, a straightforward consequence of ε-almost orthogonality is that we can find a QR-factorization of the matrix of such vectors where R (an upper diagonal matrix) has diagonal entries close to 1 and entries above the diagonal are close to 0.

Lemma C.3. *Let $u_1, \cdots, u_t \in \mathbb{R}^d$ be a set of unit vectors such that for any pair of $i < j \le t$, $|\langle u_i, u_j\rangle| \le \varepsilon$ where $\varepsilon = O(t^{-2})$. There exists an orthonormal basis $e_1, \cdots, e_t$ for the subspace spanned by $u_1, \cdots, u_t$ such that for each $i \le t$, $u_i = \sum_{j=1}^{i} a_{i,j} e_j$ where $a_{i,i}^2 \ge 1 - \sum_{j=1}^{i-1} j^2 \cdot \varepsilon^2$ and for each $j < i$, $a_{i,j}^2 \le j^2\varepsilon^2$.*

Proof: We follow the Gram-Schmidt process to construct the orthonormal basis $e_1, \cdots, e_t$ of the space spanned by $u_1, \cdots, u_t$. by first setting $e_1 = u_1$ and then processing $u_2, \cdots, u_t$, one by one.

The proof is by induction. We show that once the first j vectors $u_1, \cdots, u_j$ are processed the statement of the lemma holds for these vectors. Note that the base case of the induction trivially holds as $u_1 = e_1$. Next, suppose that the induction hypothesis holds for the first ℓ vectors $u_1, \cdots, u_\ell$.

Claim C.4. *For each $j \le \ell$, $a^2_{\ell+1,j} \le j^2\varepsilon^2$.*

Proof: The proof of the claim is itself by induction. Note that, for $j = 1$ and using the fact that $|\langle u_1, u_{\ell+1}\rangle| \le \varepsilon$, the statement holds and $a^2_{\ell+1,1} \le \varepsilon^2$. Next, suppose that the statement holds for all $j \le i < \ell$, then by $|\langle u_{i+1}, u_{\ell+1}\rangle| \le \varepsilon$,

$$\begin{aligned}
|a_{\ell+1,i+1}| &\le (|\langle u_{\ell+1}, u_{i+1}| + \sum_{j=1}^{i} |a_{\ell+1,j}| \cdot |a_{i+1,j}|)/|a_{i+1,i+1}| \\
&\le (\varepsilon + \sum_{j=1}^{i} j^2\varepsilon^2)/|a_{i+1,i+1}| \quad \triangleright \text{ by induction hypothesis on } a_{\ell+1,j} \text{ for } j \le i \\
&\le (\varepsilon + \sum_{j=1}^{i} j^2\varepsilon^2)/(1 - \sum_{j=1}^{i} j^2 \cdot \varepsilon^2)^{1/2} \quad \triangleright \text{ by induction hypothesis on } a_{i+1,i+1} \\
&\le (\varepsilon + \sum_{j=1}^{i} j^2\varepsilon^2) \cdot (1 - \sum_{j=1}^{i} j^2 \cdot \varepsilon^2)^{1/2} \cdot (1 + 2 \cdot \sum_{j=1}^{i} j^2\varepsilon^2) \\
&\le (\varepsilon + \sum_{j=1}^{i} j^2\varepsilon^2) \cdot (1 + 2 \cdot \sum_{j=1}^{i} j^2\varepsilon^2) \\
&\le \varepsilon((\sum_{j=1}^{i} j^2\varepsilon) \cdot (1 + 4\varepsilon \cdot \sum_{j=1}^{i} j^2\varepsilon) + 1) \\
&\le \varepsilon(i+1) \quad \triangleright \text{ by } \varepsilon = O(t^{-2})
\end{aligned}$$

□

Finally, since $\|u_{\ell+1}\|_2^2 = 1$, $a^2_{\ell+1,\ell+1} \ge 1 - \sum_{j=1}^{\ell} j^2\varepsilon^2$. □

Corollary C.5. *Suppose that $\overline{\varepsilon} = O(t^{-2})$. There exists an orthonormal basis $e_1, \cdots, e_t$ for the space spanned by the randomly picked vectors $v_1, \cdots, v_t$, of unit norm, so that for each i, $v_i = \sum_{j=1}^{i} a_{i,j}e_j$ where $a^2_{i,i} \ge 1 - \sum_{j=1}^{i-1} j^2 \cdot \overline{\varepsilon}^2$ and for each $j < i$, $a^2_{i,j} \le j^2 \cdot \overline{\varepsilon}^2$.*

Proof: The proof follows from Lemma C.3 and the fact that the set of vectors $v_1, \cdots, v_t$ are $\overline{\varepsilon}$-almost orthogonal (by Observation C.1). □

The first main step is to show that the greedy algorithm (with non-increasing row norm ordering) will isolate rows into their own bins until all bins are filled. In particular, this means that the heavy rows (the first to be processed) will all be isolated.

We note that because we set $\text{rank}(SA) = k$, the k-rank approximation cost is the simplified expression $\left\|AVV^\top - A\right\|_F^2$, where $U\Sigma V^\top = SA$, rather than $\left\|[AV]_k V^\top - A\right\|_F^2$. This is just the projection cost onto $row(SA)$. Also, we observe that minimizing this projection cost is the same as maximizing the sum of squared projection coefficients:

$$\begin{aligned}
\min_S \left\|A - AVV^\top\right\|_F^2 &\sim \min_S \sum_{i\in[n]} \|A_i - (\langle A_i, v_1\rangle v_1 + \ldots + \langle A_i, v_k\rangle v_k)\|_2^2 \\
&\sim \min_S \sum_{i\in[n]} (\|A_i\|_2^2 - \sum_{j\in[k]} \langle A_i, v_j\rangle^2)
\end{aligned}$$

$$\sim \max_S \sum_{i\in[n]} \sum_{j\in[k]} \langle A_i, v_j\rangle^2$$

In the following sections, we will prove that our greedy algorithm makes certain choices by showing that these choices maximize the sum of squared projection coefficients.

Lemma C.6. *For any matrix A or batch of matrices $\mathcal{A}$, at the end of iteration k, the learned CountSketch matrix S maps each row to an isolated bin. In particular, heavy rows are mapped to isolated bins.*

Proof: For any iteration $i \le k$, we consider the choice of assigning A_i to an empty bin versus an occupied bin. Without loss of generality, let this occupied bin be b_{i-1}, which already contains A_{i-1}.

We consider the difference in cost for empty versus occupied. We will do this cost comparison for A_j with $j \le i-2$, $j \ge i+1$, and finally, $j \in \{i-1, i\}$.

First, we let $\{e_1, \ldots, e_i\}$ be an orthonormal basis for $\{A_1, \ldots, A_i\}$ such that for each $r \le i$, $A_r = \sum_{j=1}^{r} a_{r,j} e_j$ where $a_{r,r} > 0$. This exists by Lemma C.3. Let $\{e_1, \ldots, e_{i-2}, \overline{e}\}$ be an orthonormal basis for $\{A_1, \ldots, A_{i+2}, A_{i-1} \pm A_i\}$. Now, $\overline{e} = c_0 e_{i-1} + c_1 e_i$ for some c_0, c_1 because $(A_{i-1} \pm A_i) - \text{proj}_{\{e_1,\ldots,e_{i-2}\}}(A_{i-1} \pm A_i) \in \text{span}(e_{i-1}, e_i)$. We note that $c_0^2 + c_1^2 = 1$ because we let $\overline{e}$ be a unit vector. We can find c_0, c_1 to be:

$$c_0 = \frac{a_{i-1,i-1} + a_{i,i-1}}{\sqrt{(a_{i-1,i-1} + a_{i,i-1})^2 + a_{i,i}^2}}, \quad c_1 = \frac{a_{i,i}}{\sqrt{(a_{i-1,i-1} + a_{i,i-1})^2 + a_{i,i}^2}}$$

1. $j \le i-2$: The cost is zero for both cases because $A_j \in \text{span}(\{e_1, \ldots, e_{i-2}\})$.

2. $j \ge i+1$: We compare the rewards (sum of squared projection coefficients) and find that $\{e_1, \ldots, e_{i-2}, \overline{e}\}$ is no better than $\{e_1, \ldots, e_i\}$.

$$\begin{aligned}
\langle A_j, \overline{e}\rangle^2 &= (c_0\langle A_j, e_{i-1}\rangle + c_1\langle A_j, e_i\rangle)^2 \\
&\le (c_1^2 + c_0^2)(\langle A_j, e_{i-1}\rangle^2 + \langle A_j, e_i\rangle^2) && \triangleright \text{Cauchy-Schwarz inequality} \\
&= \langle A_j, e_{i-1}\rangle^2 + \langle A_j, e_i\rangle^2
\end{aligned}$$

3. $j \in \{i-1, i\}$: We compute the sum of squared projection coefficients of A_{i-1} and A_i onto $\overline{e}$:

$$\begin{aligned}
&(\frac{1}{(a_{i-1,i-1} + a_{i,i-1})^2 + a_{i,i}^2}) \cdot (a_{i-1,i-1}^2(a_{i-1,i-1} + a_{i,i-1})^2 + (a_{i,i-1}(a_{i-1,i-1} + a_{i,i-1}) + a_{i,i}a_{i,i})^2) \\
=&(\frac{1}{(a_{i-1,i-1} + a_{i,i-1})^2 + a_{i,i}^2}) \cdot ((a_{i-1,i-1} + a_{i,i-1})^2(a_{i-1,i-1}^2 + a_{i,i-1}^2) + a_{i,i}^4 + 2a_{i,i-1}a_{i,i}^2(a_{i-1,i-1} + a_{i,i-1}))
\end{aligned} \tag{C.2}$$

On the other hand, the sum of squared projection coefficients of A_{i-1} and A_i onto $e_{i-1} \cup e_i$ is:

$$(\frac{(a_{i-1,i-1} + a_{i,i-1})^2 + a_{i,i}^2}{(a_{i-1,i-1} + a_{i,i-1})^2 + a_{i,i}^2}) \cdot (a_{i-1,i-1}^2 + a_{i,i-1}^2 + a_{i,i}^2) \tag{C.3}$$

Hence, the difference between the sum of squared projections of A_{i-1} and A_i onto $\overline{e}$ and $e_{i-1} \cup e_i$ is ((C.3) - (C.2))

$$\frac{a_{i,i}^2((a_{i-1,i-1}+a_{i,i-1})^2 + a_{i-1,i-1}^2 + a_{i,i-1}^2 - 2a_{i,i-1}(a_{i-1,i-1}+a_{i,i-1}))}{((a_{i-1,i-1}+a_{i,i-1})^2 + a_{i,i}^2)}$$
$$= \frac{2a_{i,i}^2 a_{i-1,i-1}^2}{((a_{i-1,i-1}+a_{i,i-1})^2 + a_{i,i}^2)} > 0$$

Thus, we find that $\{e_1, \ldots, e_i\}$ is a strictly better basis than $\{e_1, \ldots, e_{i-2}, \overline{e}\}$. This means the greedy algorithm will choose to place A_i in an empty bin. □

Next, we show that none of the rows left to be processed (all light rows) will be assigned to the same bin as a heavy row. The main proof idea is to compare the cost of "colliding" with a heavy row to the cost of "avoiding" the heavy rows. Specifically, we compare the *decrease* (before and after bin assignment of a light row) in the sum of squared projection coefficients, lower-bounding it in the former case and upper-bounding it in the latter.

We introduce some results that will be used in Lemma C.10.

Claim C.7. *Let $A_{k+r}, r \in [1, \ldots, n-k]$ be a light row not yet processed by the greedy algorithm. Let $\{e_1, \ldots, e_k\}$ be the Gram-Schmidt basis for the current $\{w_1, \ldots, w_k\}$. Let $\beta = O(n^{-1}k^{-3})$ upper bound the inner products of normalized $A_{k+r}, w_1, \ldots, w_k$. Then, for any bin i, $\langle e_i, A_{k+r}\rangle^2 \leq \beta^2 \cdot k^2$.*

Proof: This is a straightforward application of Lemma C.3. From that, we have $\langle A_{k+r}, e_i\rangle^2 \leq i^2\beta^2$, for $i \in [1, \ldots, k]$, which means $\langle A_{k+r}, e_i\rangle^2 \leq k^2\beta^2$. □

Claim C.8. *Let A_{k+r} be a light row that has been processed by the greedy algorithm. Let $\{e_1, \ldots, e_k\}$ be the Gram-Schmidt basis for the current $\{w_1, \ldots, w_k\}$. If A_{k+r} is assigned to bin b_{k-1} (w.l.o.g.), the squared projection coefficient of A_{k+r} onto $e_i, i \neq k-1$ is at most $4\beta^2 \cdot k^2$, where $\beta = O(n^{-1}k^{-3})$ upper bounds the inner products of normalized $A_{k+r}, w_1, \cdots, w_k$.*

Proof: Without loss of generality, it suffices to bound the squared projection of A_{k+r} onto the direction of w_k that is orthogonal to the subspace spanned by $w_1, \cdots, w_{k-1}$. Let $e_1, \cdots, e_k$ be an orthonormal basis of $w_1, \cdots, w_k$ guaranteed by Lemma C.3. Next, we expand the orthonormal basis to include e_{k+1} so that we can write the normalized vector of A_{k+r} as $v_{k+r} = \sum_{j=1}^{k+1} b_j e_j$. By a similar approach to the proof of Lemma C.3, for each $j \leq k-2$, $b_j \leq \beta^2 j^2$. Next, since $|\langle w_k, v_{k+r}\rangle| \leq \beta$,

$$|b_k| \leq \frac{1}{|\langle w_k, e_k\rangle|} \cdot (|\langle w_k, v_{k+r}\rangle| + \sum_{j=1}^{k-1} |b_j \cdot \langle w_k, e_j\rangle|)$$
$$\leq \frac{1}{\sqrt{1 - \sum_{j=1}^{k-1} \beta^2 \cdot j^2}} \cdot (\beta + \sum_{j=1}^{k-2} \beta^2 \cdot j^2 + (k-1)\cdot\beta) \quad \triangleright\, |b_{k-1}| \leq 1$$
$$= \frac{\beta + \sum_{j=1}^{k-2} \beta^2 \cdot j^2}{\sqrt{1 - \sum_{j=1}^{k-1} \beta^2 \cdot j^2}} + (k-1)\beta$$

$$\leq 2(k-1)\beta - \frac{\beta^2(k-1)^2}{\sqrt{1-\sum_{j=1}^{k-1}\beta^2\cdot j^2}} \qquad \triangleright \text{similar to the proof of Lemma C.3}$$
$$< 2\beta\cdot k$$

Hence, the squared projection of A_{k+r} onto e_k is at most $4\beta^2\cdot k^2\cdot\|A_{k+r}\|_2^2$. We assumed $\|A_{k+r}\|$.□

Claim C.9. *We assume that the absolute values of the inner products of vectors in $v_1,\cdots,v_n$ are at most $\overline{\varepsilon} < 1/(n^2\sum_{A_i\in b}\|A_i\|_2)$ and the absolute values of the inner products of the normalized vectors of $w_1,\cdots,w_k$ are at most $\beta = O(n^{-3}k^{-\frac{3}{2}})$. Suppose that bin b contains the row A_{k+r}. Then, the squared projection of A_{k+r} onto the direction of w orthogonal to $\mathrm{span}(\{w_1,\cdots,w_k\}\setminus\{w\})$ is at most $\frac{\|A_{k+r}\|_2^4}{\|w\|_2^2}+O(n^{-2})$ and is at least $\frac{\|A_{k+r}\|_2^4}{\|w\|_2^2}-O(n^{-2})$.*

Proof: Without loss of generality, we assume that A_{k+r} is mapped to b_k; $w = w_k$. First, we provide an upper and a lower bound for $|\langle v_{k+r},\overline{w}_k\rangle|$ where for each $i\leq k$, we let $\overline{w}_i = \frac{w_i}{\|w_i\|_2}$ denote the normalized vector of w_i. Recall that by definition $v_{k+r} = \frac{A_{k+r}}{\|A_{k+r}\|_2}$.

$$\begin{aligned}
|\langle\overline{w}_k, v_{k+r}\rangle| &\leq \frac{\|A_{k+r}\|_2 + \sum_{A_i\in b_k}\overline{\varepsilon}\,\|A_i\|_2}{\|w_k\|_2} \\
&\leq \frac{\|A_{k+r}\|_2 + n^{-2}}{\|w_k\|_2} && \triangleright \text{ by } \overline{\varepsilon} < \frac{n^{-2}}{\sum_{A_i\in b_k}\|A_i\|_2} \\
&\leq \frac{\|A_{k+r}\|_2}{\|w_k\|_2} + n^{-2} && \triangleright \|w_k\|_2 \geq 1 \qquad \text{(C.4)}
\end{aligned}$$

$$\begin{aligned}
|\langle\overline{w}_k, v_{k+r}\rangle| &\geq \frac{\|A_{k+r}\|_2 - \sum_{A_i\in b_k}\|A_i\|_2\cdot\overline{\varepsilon}}{\|w_k\|_2} \\
&\geq \frac{\|A_{k+r}\|_2}{\|w_k\|_2} - n^{-2} \qquad \text{(C.5)}
\end{aligned}$$

Now, let $\{e_1,\cdots,e_k\}$ be an orthonormal basis for the subspace spanned by $\{w_1,\cdots,w_k\}$ guaranteed by Lemma C.3. Next, we expand the orthonormal basis to include e_{k+1} so that we can write $v_{k+r} = \sum_{j=1}^{k+1} b_j e_j$. By a similar approach to the proof of Lemma C.3, we can show that for each $j\leq k-1$, $b_j^2\leq\beta^2 j^2$. Moreover,

$$\begin{aligned}
|b_k| &\leq \frac{1}{|\langle\overline{w}_k, e_k\rangle|}\cdot\Big(|\langle\overline{w}_k, v_{k+r}\rangle| + \sum_{j=1}^{k-1}|b_j\cdot\langle\overline{w}_k, e_j\rangle|\Big) \\
&\leq \frac{1}{\sqrt{1-\sum_{j=1}^{k-1}\beta^2\cdot j^2}}\cdot\Big(|\langle\overline{w}_k, v_{k+r}\rangle| + \sum_{j=1}^{k-1}\beta^2\cdot j^2\Big) && \triangleright \text{ by Lemma C.3} \\
&\leq \frac{1}{\sqrt{1-\sum_{j=1}^{k-1}\beta^2\cdot j^2}}\cdot\Big(n^{-2} + \frac{\|A_{k+r}\|_2}{\|w_k\|_2} + \sum_{j=1}^{k-1}\beta^2\cdot j^2\Big) && \triangleright \text{ by (C.4)}
\end{aligned}$$

$$
\begin{aligned}
&< \beta \cdot k + \frac{1}{\sqrt{1-\beta^2 k^3}} \cdot (n^{-2} + \frac{\|A_{k+r}\|_2}{\|w_k\|_2}) && \triangleright \text{similar to the proof of Lemma C.3} \\
&\leq O(n^{-2}) + (1 + O(n^{-2})) \frac{\|A_{k+r}\|_2}{\|w_k\|_2} && \triangleright \text{by } \beta = O(n^{-3}k^{-\frac{3}{2}}) \\
&\leq \frac{\|A_{k+r}\|_2}{\|w_k\|_2} + O(n^{-2}) && \triangleright \frac{\|A_{k+r}\|_2}{\|w_k\|_2} \leq 1
\end{aligned}
$$

and,

$$
\begin{aligned}
|b_k| &\geq \frac{1}{|\langle \overline{w}_k, e_k \rangle|} \cdot (|\langle \overline{w}_k, v_{k+r} \rangle| - \sum_{j=1}^{k-1} |b_j \cdot \langle \overline{w}_k, e_j \rangle|) \\
&\geq |\langle \overline{w}_k, v_{k+r} \rangle| - \sum_{j=1}^{k-1} \beta^2 \cdot j^2 && \triangleright \text{since } |\langle \overline{w}_k, e_k \rangle| \leq 1 \\
&\geq \frac{\|A_{k+r}\|_2}{\|w_k\|_2} - n^{-2} - \sum_{j=1}^{k-1} \beta^2 \cdot j^2 && \triangleright \text{by (C.5)} \\
&\geq \frac{\|A_{k+r}\|_2}{\|w_k\|_2} - O(n^{-2}) && \triangleright \text{by } \beta = O(n^{-3}k^{-\frac{3}{2}})
\end{aligned}
$$

Hence, the squared projection of A_{k+r} onto e_k is at most $\frac{\|A_{k+r}\|_2^4}{\|w_k\|_2^2} + O(n^{-2})$ and is at least $\frac{\|A_{k+r}\|_2^4}{\|w_k\|_2^2} - O(n^{-2})$. □

Now, we show that at the end of the algorithm no light row will be assigned to the bins that contain heavy rows.

Lemma C.10. *We assume that the absolute values of the inner products of vectors in $v_1, \cdots, v_n$ are at most $\overline{\varepsilon} < \min\{n^{-2}k^{-\frac{5}{3}}, (n \sum_{A_i \in w} \|A_i\|_2)^{-1}\}$. At each iteration $k + r$, the greedy algorithm will assign the light row A_{k+r} to a bin that does not contain a heavy row.*

Proof: The proof is by induction. Lemma C.6 implies that no light row has been mapped to a bin that contains a heavy row for the first k iterations. Next, we assume that this holds for the first $k + r - 1$ iterations and show that is also must hold for the $k + r^{\text{th}}$ iteration.

To this end, we compare the sum of squared projection coefficients when A_{k+r} avoids and collides with a heavy row.

First, we upper bound $\beta = \max_{i \neq j \leq k} |\langle w_i, w_j \rangle| / (\|w_i\|_2 \|w_j\|_2)$. Let c_i and c_j respectively denote the number of rows assigned to b_i and b_j.

$$
\begin{aligned}
\beta = \max_{i \neq j \leq k} \frac{|\langle w_i, w_j \rangle|}{\|w_i\|_2 \|w_j\|_2} &\leq \frac{c_i \cdot c_j \cdot \overline{\varepsilon}}{\sqrt{c_i - 2\overline{\varepsilon} c_i^2} \cdot \sqrt{c_j - 2\overline{\varepsilon} c_j^2}} && \triangleright \text{Observation C.2} \\
&\leq 16 \overline{\varepsilon} \sqrt{c_i c_j} && \triangleright \overline{\varepsilon} \leq n^{-2}k^{-5/3} \\
&\leq n^{-1} k^{-\frac{5}{3}} && \triangleright \overline{\varepsilon} \leq n^{-2}k^{-5/3}
\end{aligned}
$$

1. If A_{k+r} is assigned to a bin that contains c light rows and no heavy rows. In this case, the projection loss of the heavy rows $A_1, \cdots, A_s$ onto $row(SA)$ remains zero. Thus, we only need to bound the change in the sum of squared projection coefficients of the light rows before and after iteration $k+r$.

Without loss of generality, let w_k denote the bin that contains A_{k+r}. Since $\mathcal{S}_{k-1} = \text{span}(\{w_1, \cdots, w_{k-1}\})$ has not changed, we only need to bound the difference in cost between projecting onto the component of $w_k - A_{k+r}$ orthogonal to $\mathcal{S}_{k-1}$ and the component of w_k orthogonal to $\mathcal{S}_{k-1}$, respectively denoted as e_k and $\overline{e}_k$.

I. By Claim C.7, for the light rows that are not yet processed (i.e., A_j for $j > k+r$), the squared projection of each onto e_k is at most $\beta^2 k^2$. Hence, the total decrease in the squared projection is at most $(n-k-r) \cdot \beta^2 k^2$.

II. By Claim C.8, for the processed light rows that are not mapped to the last bin, the squared projection of each onto e_k is at most $4\beta^2 k^2$. Hence, the total decrease in the squared projection cost is at most $(r-1) \cdot 4\beta^2 k^2$.

III. For each row $A_i \neq A_{k+r}$ that is mapped to the last bin, by Claim C.9 and the fact $\|A_i\|_2^4 = \|A_i\|_2^2 = 1$, the squared projection of A_i onto e_k is at most $\frac{\|A_i\|_2^2}{\|w_k - A_{k+r}\|_2^2} + O(n^{-2})$ and the squared projection of A_i onto $\overline{e}_k$ is at least $\frac{\|A_i\|_2^2}{\|w_k\|_2^2} - O(n^{-2})$.

Moreover, the squared projection of A_{k+r} onto e_k compared to $\overline{e}_k$ increases by at least $(\frac{\|A_{k+r}\|_2^2}{\|w_k\|_2^2} - O(n^{-2})) - O(n^{-2}) = \frac{\|A_{k+r}\|_2^2}{\|w_k\|_2^2} - O(n^{-2})$.

Hence, the total squared projection of the rows in the bin b_k decreases by at least:

$$
\begin{aligned}
&(\sum_{A_i \in w_k / \{A_{r+k}\}} \frac{\|A_i\|_2^2}{\|w_k - A_{r+k}\|_2^2} + O(n^{-2})) - (\sum_{A_i \in w_k} \frac{\|A_i\|_2^2}{\|w_k\|_2^2} - O(n^{-2})) \\
\leq &\frac{\|w_k - A_{r+k}\|_2^2 + O(n^{-1})}{\|w_k - A_{r+k}\|_2^2} - \frac{\|w_k\|_2^2 - O(n^{-1})}{\|w_k\|_2^2} + O(n^{-1}) \qquad \triangleright \text{ by Observation C.2} \\
\leq &O(n^{-1})
\end{aligned}
$$

Hence, summing up the bounds in items I to III above, the total decrease in the sum of squared projection coefficients is at most $O(n^{-1})$.

2. If A_{k+r} is assigned to a bin that contains a heavy row. Without loss of generality, we can assume that A_{k+r} is mapped to b_k that contains the heavy row A_s. In this case, the distance of heavy rows $A_1, \cdots, A_{s-1}$ onto the space spanned by the rows of SA is zero. Next, we bound the amount of change in the squared distance of A_s and light rows onto the space spanned by the rows of SA.

Note that the $(k-1)$-dimensional space corresponding to $w_1, \cdots, w_{k-1}$ has not changed. Hence, we only need to bound the decrease in the projection distance of A_{k+r} onto $\overline{e}_k$ compared to e_k (where $\overline{e}_k, e_k$ are defined similarly as in the last part).

1. For the light rows other than A_{k+r}, the squared projection of each onto e_k is at most $\beta^2 k^2$. Hence, the total increase in the squared projection of light rows onto e_k is at most $(n-k) \cdot \beta^2 k^2 = O(n^{-1})$.

2. By Claim C.9, the sum of squared projections of A_s and A_{k+r} onto e_k decreases by at least

$$\begin{aligned}
&\|A_s\|_2^2 - \left(\frac{\|A_s\|_2^4 + \|A_{k+r}\|_2^4}{\|A_s + A_{r+k}\|_2^2} + O(n^{-1})\right) \\
\geq &\|A_s\|_2^2 - \left(\frac{\|A_s\|_2^4 + \|A_{k+r}\|_2^4}{\|A_s\|_2^2 + \|A_{r+k}\|_2^2 - n^{-O(1)}} + O(n^{-1})\right) && \triangleright \text{ by Observation C.2} \\
\geq &\frac{\|A_{r+k}\|_2^2 \left(\|A_s\|_2^2 - \|A_{k+r}\|_2^2\right) - \|A_s\|_2^2 \cdot O(n^{-1})}{\|A_s\|_2^2 + \|A_{r+k}\|_2^2 - O(n^{-1})} - O(n^{-1}) \\
\geq &\frac{\|A_{r+k}\|_2^2 \left(\|A_s\|_2^2 - \|A_{k+r}\|_2^2\right) - \|A_s\|_2^2 \cdot O(n^{-1})}{\|A_s\|_2^2 + \|A_{r+k}\|_2^2} - O(n^{-1}) \\
\geq &\frac{\|A_{r+k}\|_2^2 \left(\|A_s\|_2^2 - \|A_{k+r}\|_2^2\right)}{\|A_s\|_2^2 + \|A_{r+k}\|_2^2} - O(n^{-1}) \\
\geq &\frac{\|A_{r+k}\|_2^2 \left(1 - \left(\|A_{k+r}\|_2^2 / \|A_s\|_2^2\right)\right)}{1 + \left(\|A_{r+k}\|_2^2 / \|A_s\|_2^2\right)} - O(n^{-1}) \\
\geq &\|A_{r+k}\|_2^2 \left(1 - \frac{\|A_{k+r}\|_2}{\|A_s\|_2}\right) - O(n^{-1}) && \triangleright \frac{1-\varepsilon^2}{1+\varepsilon^2} \geq 1 - \varepsilon
\end{aligned}$$

Hence, in this case, the total decrease in the squared projection is at least

$$\begin{aligned}
\|A_{r+k}\|_2^2 \left(1 - \frac{\|A_{k+r}\|_2}{\|A_s\|_2}\right) - O(n^{-1}) &= 1 - \frac{\|A_{k+r}\|_2}{\|A_s\|_2}) - O(n^{-1}) && \triangleright \|A_{r+k}\|_2 = 1 \\
&= 1 - (1/\sqrt{\ell}) - O(n^{-1}) && \triangleright \|A_s\|_2 = \sqrt{\ell}
\end{aligned}$$

Thus, for a sufficiently large value of ℓ, the greedy algorithm will assign A_{k+r} to a bin that only contains light rows. This completes the inductive proof and in particular implies that at the end of the algorithm, heavy rows are assigned to isolated bins. □

Corollary C.11. *The approximation loss of the best* rank-k *approximate solution in the rowspace $S_g A$ for $A \sim \mathcal{A}_{sp}(s, \ell)$ where $A \in \mathbb{R}^{n \times d}$ for $d = \Omega(n^4 k^4 \log n)$ and S_g is the CountSketch constructed by the greedy algorithm with non-increasing order is at most $n - s$.*

Proof: First, we need to show that absolute values of the inner products of vectors in $v_1, \cdots, v_n$ is at most $\overline{\varepsilon} < \min\{n^{-2}k^{-2}, (n \sum_{A_i \in w} \|A_i\|_2)^{-1}\}$ so that we can apply Lemma C.10. To show this, note that by Observation C.1, $\overline{\varepsilon} \leq 2\sqrt{\frac{\log n}{d}} \leq n^{-2}k^{-2}$ since $d = \Omega(n^4 k^4 \log n)$. The proof follows from Lemma C.6 and Lemma C.10. Since all heavy rows are mapped to isolated bins, the projection loss of the light rows is at most $n - s$. □

Next, we bound the Frobenius norm error of the best rank-k-approximation solution constructed by the standard CountSketch with a randomly chosen sparsity pattern.

Lemma C.12. *Let $s = \alpha k$ where $0.7 < \alpha < 1$. The expected squared loss of the best* rank-k *approximate solution in the rowspace $S_r A$ for $A \in \mathbb{R}^{n \times d} \sim \mathcal{A}_{sp}(s, \ell)$ where $d = \Omega(n^6 \ell^2)$ and S_r is the sparsity pattern of CountSketch is chosen uniformly at random is at least $n + \frac{\ell k}{4e} - (1+\alpha)k - n^{-O(1)}$.*

Proof: We can interpret the randomized construction of the CountSketch as a "balls and bins" experiment. In particular, considering the heavy rows, we compute the expected number of bins (i.e., rows in S_rA) that contain a heavy row. Note that the expected number of rows in S_rA that do not contain any heavy row is $k \cdot (1 - \frac{1}{k})^s \geq k \cdot e^{-\frac{s}{k-1}}$. Hence, the number of rows in S_rA that contain a heavy row of A is at most $k(1 - e^{-\frac{s}{k-1}})$. Thus, at least $s - k(1 - e^{-\frac{s}{k-1}})$ heavy rows are not mapped to an isolated bin (i.e., they collide with some other heavy rows). Then, it is straightforward to show that the squared loss of each such row is at least $\ell - n^{-O(1)}$.

Claim C.13. *Suppose that heavy rows $A_{r_1}, \cdots, A_{r_c}$ are mapped to the same bin via a CountSketch S. Then, the total squared distances of these rows from the subspace spanned by SA is at least $(c-1)\ell - O(n^{-1})$.*

Proof: Let b denote the bin that contains the rows $A_{r_1}, \cdots, A_{r_c}$ and suppose that it has c' light rows as well. Note that by Claim C.8 and Claim C.9, the squared projection of each row A_{r_i} onto the subspace spanned by the k bins is at most

$$\begin{aligned}
&\frac{\|A_{h_i}\|_2^4}{\|w\|_2^2} + O(n^{-1}) \\
\leq &\frac{\ell^2}{c\ell + c' - 2\overline{\varepsilon}(c^2\ell + cc'\sqrt{\ell} + c'^2)} + O(n^{-1}) \\
\leq &\frac{\ell^2}{c\ell - n^{-O(1)}} + n^{-O(1)} && \triangleright \text{ by } \overline{\varepsilon} \leq n^{-3}\ell^{-1} \\
\leq &\frac{\ell^2}{c^2\ell^2} \cdot (c\ell + O(n^{-1}) + O(n^{-1}) \\
\leq &\frac{\ell}{c} + O(n^{-1})
\end{aligned}$$

Hence, the total squared loss of these c heavy rows is at least $c\ell - c\cdot(\frac{\ell}{c} + O(n^{-1})) \geq (c-1)\ell - O(n^{-1})$.□

Hence, the expected total squared loss of heavy rows is at least:

$$\begin{aligned}
&\ell \cdot (s - k(1 - e^{-\frac{s}{k-1}})) - s \cdot n^{-O(1)} \\
\geq &\ell \cdot k(\alpha - 1 + e^{-\alpha}) - \ell\alpha - n^{-O(1)} && \triangleright\ s = \alpha \cdot (k-1) \text{ where } 0.7 < \alpha < 1 \\
\geq &\frac{\ell k}{2e} - \ell - n^{-O(1)} && \triangleright\ \alpha \geq 0.7 \\
\geq &\frac{\ell k}{4e} - O(n^{-1}) && \triangleright \text{ assuming } k > 4e
\end{aligned}$$

Next, we compute a lower bound on the expected squared loss of the light rows. Note that Claim C.8 and Claim C.9 imply that when a light row collides with other rows, its contribution to the total squared loss (which the loss accounts for the amount it decreases from the squared projection of the other rows in the bin as well) is at least $1 - O(n^{-1})$. Hence, the expected total squared loss of the light rows is at least:

$$(n - s - k)(1 - O(n^{-1})) \geq (n - (1+\alpha) \cdot k) - O(n^{-1})$$

Hence, the expected squared loss of a CountSketch whose sparsity is picked at random is at least

$$\frac{\ell k}{4e} - O(n^{-1}) + n - (1+\alpha)k - O(n^{-1}) \geq n + \frac{\ell k}{4e} - (1+\alpha)k - O(n^{-1}) \qquad \square$$

Corollary C.14. *Let $s = \alpha(k-1)$ where $0.7 < \alpha < 1$ and let $\ell \geq \frac{(4e+1)n}{\alpha k}$. Let S_g be the CountSketch whose sparsity pattern is learned over a training set drawn from $\mathcal{A}_{sp}$ via the greedy approach. Let S_r be a CountSketch whose sparsity pattern is picked uniformly at random. Then, for an $n \times d$ matrix $A \sim \mathcal{A}_{sp}$ where $d = \Omega(n^6\ell^2)$, the expected loss of the best* rank-k *approximation of A returned by S_r is worse than the approximation loss of the best* rank-k *approximation of A returned by S_g by at least a constant factor.*

Proof:

$$\begin{aligned}
\mathbf{E}_{S_r}[\min_{\text{rank-}k\ X \in \text{rowsp}(S_r A)} \|X - A\|_F^2] &\geq n + \frac{\ell k}{4e} - (1+\alpha)k - n^{-O(1)} && \triangleright \text{ Lemma C.12} \\
&\geq (1 + 1/\alpha)(n - s) && \triangleright\ \ell \geq \frac{(4e+1)n}{\alpha k} \\
&= (1 + 1/\alpha) \min_{\text{rank-}k\ X \in \text{rowsp}(S_g A)} \|X - A\|_F^2 && \triangleright \text{ Corollary C.11}
\end{aligned}$$

$\square$

C.2 Zipfian on squared row norms.

Each matrix $A \in \mathbb{R}^{n \times d} \sim \mathcal{A}_{zipf}$ has rows which are uniformly random and orthogonal. Each A has 2^{i+1} rows of squared norm $n^2/2^{2i}$ for $i \in [1, \ldots, \mathcal{O}(\log(n))]$. We also assume that each row has the same squared norm for all members of $\mathcal{A}_{zipf}$.

In this section, the s rows with largest norm are called the *heavy* rows and the remaining are the *light* rows. For convenience, we number the heavy rows $1 - s$; however, the heavy rows can appear at any indices, as long as any row of a given index has the same norm for all members of $\mathcal{A}_{zipf}$. Also, we assume that $s \leq k/2$ and, for simplicity, $s = \sum_{i=1}^{h_s} 2^{i+1}$ for some $h_s \in \mathbb{Z}^+$. That means the minimum squared norm of a heavy row is $n^2/2^{2h_s}$ and the maximum squared norm of a light row is $n^2/2^{2h_s+2}$.

The analysis of the greedy algorithm ordered by non-increasing row norms on this family of matrices is similar to our analysis for the spiked covariance model. Here we analyze the case in which rows are orthogonal. By continuity, if the rows are close enough to being orthogonal, all decisions made by the greedy algorithm will be the same.

As a first step, by Lemma C.6, at the end of iteration k the first k rows are assigned to different bins. Then, via a similar inductive proof, we show that none of the light rows are mapped to a bin that contains one of the top s heavy rows.

Lemma C.15. *At each iteration $k+r$, the greedy algorithm picks the position of the non-zero value in the $(k+r)$-th column of the CountSketch matrix S so that the light row A_{k+r} is mapped to a bin that does not contain any of top s heavy rows.*

Proof: We prove the statement by induction. The base case $r = 0$ trivially holds as the first k rows are assigned to distinct bins. Next we assume that in none of the first $k + r - 1$ iterations a light row is assigned to a bin that contains a heavy row. Now, we consider the following cases:

1. If A_{k+r} is assigned to a bin that only contains light rows. Without loss of generality we can assume that A_{k+r} is assigned to b_k. Since the vectors are orthogonal, we only need to bound the difference in the projection of A_{k+r} and the light rows that are assigned to b_k onto the direction of w_k before and after adding A_{k+r} to b_k. In this case, the total squared loss corresponding to rows in b_k and A_{k+r} before and after adding A_{k+1} are respectively

$$\textit{before adding } A_{k+r} \textit{ to } b_k\textit{:}\ \ \|A_{k+r}\|_2^2 + \sum_{A_j \in b_k} \|A_j\|_2^2 - \left(\frac{\sum_{A_j \in b_k} \|A_j\|_2^4}{\sum_{A_j \in b_k} \|A_j\|_2^2}\right)$$

$$\textit{after adding } A_{k+r} \textit{ to } b_k\textit{:}\ \ \|A_{k+r}\|_2^2 + \sum_{A_j \in b_k} \|A_j\|_2^2 - \left(\frac{\|A_{k+r}\|_2^4 + \sum_{A_j \in b_k} \|A_j\|_2^4}{\|A_{k+r}\|_2^2 + \sum_{A_j \in b_k} \|A_j\|_2^2}\right)$$

Thus, the amount of increase in the squared loss is

$$\begin{aligned}
\left(\frac{\sum_{A_j \in b_k} \|A_j\|_2^4}{\sum_{A_j \in b_k} \|A_j\|_2^2}\right) - \left(\frac{\|A_{k+r}\|_2^4 + \sum_{A_j \in b_k} \|A_j\|_2^4}{\|A_{k+r}\|_2^2 + \sum_{A_j \in b_k} \|A_j\|_2^2}\right) &= \frac{\|A_{k+r}\|_2^2 \cdot \sum_{A_j \in b_k} \|A_j\|_2^4 - \|A_{k+r}\|_2^4 \cdot \sum_{A_j \in b_k} \|A_j\|_2^2}{(\sum_{A_j \in b_k} \|A_j\|_2^2)(\|A_{k+r}\|_2^2 + \sum_{A_j \in b_k} \|A_j\|_2^2)} \\
&= \|A_{k+r}\|_2^2 \cdot \frac{\frac{\sum_{A_j \in b_k} \|A_j\|_2^4}{\sum_{A_j \in b_k} \|A_j\|_2^2} - \|A_{k+r}\|_2^2}{\sum_{A_j \in b_k} \|A_j\|_2^2 + \|A_{k+r}\|_2^2} \\
&\leq \|A_{k+r}\|_2^2 \cdot \frac{\sum_{A_j \in b_k} \|A_j\|_2^2 - \|A_{k+r}\|_2^2}{\sum_{A_j \in b_k} \|A_j\|_2^2 + \|A_{k+r}\|_2^2} \qquad \text{(C.6)}
\end{aligned}$$

2. If A_{k+r} is assigned to a bin that contains a heavy row. Without loss of generality and by the induction hypothesis, we assume that A_{k+r} is assigned to a bin b that only contains a heavy row A_j. Since the rows are orthogonal, we only need to bound the difference in the projection of A_{k+r} and A_j In this case, the total squared loss corresponding to A_j and A_{k+r} before and after adding A_{k+1} to b are respectively

$$\textit{before adding } A_{k+r} \textit{ to } b_k\textit{:}\ \ \|A_{k+r}\|_2^2$$

$$\textit{after adding } A_{k+r} \textit{ to } b_k\textit{:}\ \ \|A_{k+r}\|_2^2 + \|A_j\|_2^2 - \left(\frac{\|A_{k+r}\|_2^4 + \|A_j\|_2^4}{\|A_{k+r}\|_2^2 + \|A_j\|_2^2}\right)$$

Thus, the amount of increase in the squared loss is

$$\|A_j\|_2^2 - \left(\frac{\|A_{k+r}\|_2^4 + \|A_j\|_2^4}{\|A_{k+r}\|_2^2 + \|A_j\|_2^2}\right) = \|A_{k+r}\|_2^2 \cdot \frac{\|A_j\|_2^2 - \|A_{k+r}\|_2^2}{\|A_j\|_2^2 + \|A_{k+r}\|_2^2} \qquad \text{(C.7)}$$

Then (C.7) is larger than (C.6) if $\|A_j\|_2^2 \geq \sum_{A_i \in b_k} \|A_i\|_2^2$. Next, we show that at every inductive iteration, there exists a bin b which only contains light rows and whose squared norm is smaller than the squared norm of any heavy row. For each value m, define h_m so that $m = \sum_{i=1}^{h_m} 2^{i+1} = 2^{h_m+2} - 2$.

Recall that all heavy rows have squared norm at least $\frac{n^2}{2^{2h_s}}$. There must be a bin b that only contains light rows and has squared norm at most

$$\|w\|_2^2 = \sum_{A_i \in b} \|A_i\|_2^2 \leq \frac{n^2}{2^{2(h_s+1)}} + \frac{\sum_{i=h_k+1}^{h_n} \frac{2^{i+1} n^2}{2^{2i}}}{k-s}$$

$$
\begin{aligned}
&\leq \frac{n^2}{2^{2(h_s+1)}} + \frac{2n^2}{2^{h_k}(k-s)} \\
&\leq \frac{n^2}{2^{2(h_s+1)}} + \frac{n^2}{2^{2h_k}} && \triangleright\ s \leq k/2 \text{ and } k > 2^{h_k+1} \\
&\leq \frac{n^2}{2^{2h_s+1}} && \triangleright\ h_k \geq h_s + 1 \\
&< \|A_s\|_2^2
\end{aligned}
$$

Hence, the greedy algorithm will map A_{k+r} to a bin that only contains light rows. □

Corollary C.16. *The squared loss of the best* rank-k *approximate solution in the rowspace of $S_g A$ for $A \in \mathbb{R}^{n\times d} \sim \mathcal{A}_{zipf}$ where $A \in \mathbb{R}^{n\times d}$ and S_g is the CountSketch constructed by the greedy algorithm with non-increasing order, is $< \frac{n^2}{2^{h_k-2}}$.*

Proof: At the end of iteration k, the total squared loss is $\sum_{i=h_k+1}^{h_n} 2^{i+1} \cdot \frac{n^2}{2^{2i}}$. After that, in each iteration $k+r$, by (C.6), the squared loss increases by at most $\|A_{k+r}\|_2^2$. Hence, the total squared loss in the solution returned by S_g is at most

$$
\begin{aligned}
2\Big(\sum_{i=h_k+1}^{h_n} \frac{2^{i+1}n^2}{2^{2i}}\Big) = 4n^2 \cdot \sum_{i=h_k+1}^{h_n} \frac{1}{2^i} \\
< \frac{4n^2}{2^{h_k}} = \frac{n^2}{2^{h_k-2}}
\end{aligned}
$$

□

Next, we bound the squared loss of the best rank-k-approximate solution constructed by the standard CountSketch with a randomly chosen sparsity pattern.

Observation C.17. *Let us assume that the orthogonal rows $A_{r_1}, \cdots, A_{r_c}$ are mapped to same bin and for each $i \leq c$, $\|A_{r_1}\|_2^2 \geq \|A_{r_i}\|_2^2$. Then, the total squared loss of $A_{r_1}, \cdots, A_{r_c}$ after projecting onto $A_{r_1} \pm \cdots \pm A_{r_c}$ is at least $\|A_{r_2}\|_2^2 + \cdots + \|A_{r_c}\|_2^2$.*

Proof: Note that since $A_{r_1}, \cdots, A_{r_c}$ are orthogonal, for each $i \leq c$, the squared projection of A_{r_i} onto $A_{r_1} \pm \cdots \pm A_{r_c}$ is $\|A_{r_i}\|_2^4 / \sum_{j=1}^c \left\|A_{r_j}\right\|_2^2$. Hence, the sum of squared projection coefficients of $A_{r_1}, \cdots, A_{r_c}$ onto $A_{r_1} \pm \cdots \pm A_{r_c}$ is

$$
\frac{\sum_{j=1}^c \left\|A_{r_j}\right\|_2^4}{\sum_{j=1}^c \left\|A_{r_j}\right\|_2^2} \leq \|A_{r_1}\|_2^2
$$

Hence, the total projection loss of $A_{r_1}, \cdots, A_{r_c}$ onto $A_{r_1} \pm \cdots \pm A_{r_c}$ is at least

$$
\sum_{j=1}^{c} \left\|A_{r_j}\right\|_2^2 - \|A_{r_1}\|_2^2 = \|A_{r_2}\|_2^2 + \cdots + \|A_{r_c}\|_2^2 .
$$

□

In particular, Observation C.17 implies that whenever two rows are mapped into a same bin, the squared norm of the row with smaller norm *fully* contributes to the total squared loss of the solution.

Lemma C.18. *For $k > 2^{10} - 2$, the expected squared loss of the best* rank-k *approximate solution in the rowspace of $S_r A$ for $A_{n\times d} \sim \mathcal{A}_{zipf}$, where S_r is the sparsity pattern of a CountSketch chosen uniformly at random, is at least $\frac{1.095n^2}{2^{h_k-2}}$.*

Proof: In light of Observation C.17, we need to compute the expected number of collision between rows with "large" norm. We can interpret the randomized construction of the CountSketch as a "balls and bins" experiment.

For each $0 \le j \le h_k$, let $\mathcal{A}_j$ denote the set of rows with squared norm $\frac{n^2}{2^{2(h_k-j)}}$ and let $\mathcal{A}_{>j} = \bigcup_{j<i\le h_k} \mathcal{A}_i$. Note that for each j, $|\mathcal{A}_j| = 2^{h_k-j+1}$ and $|\mathcal{A}_{>j}| = \sum_{i=j+1}^{h_k} 2^{h_k-i+1} = \sum_{i=1}^{h_k-j} 2^i = 2(2^{h_k-j} - 1)$. Moreover, note that $k = 2(2^{h_k+1} - 1)$. Next, for a row A_r in $\mathcal{A}_j$ $(0 \le j < h_k)$, we compute the probability that at least one row in $\mathcal{A}_{>j}$ collide with A_r.

$$\begin{aligned}
\mathbf{Pr}[\text{at least one row in } \mathcal{A}_{>j} \text{ collide with } A_r] &= (1 - (1 - \frac{1}{k})^{|\mathcal{A}_{>j}|}) \\
&\ge (1 - e^{-\frac{|\mathcal{A}_{>j}|}{k}}) \\
&= (1 - e^{-\frac{2^{h_k-j}-1}{2^{h_k+1}-1}}) \\
&\ge (1 - e^{-2^{-j-2}}) \qquad \triangleright \text{since } \frac{2^{h_k-j}-1}{2^{h_k+1}-1} > 2^{-j-2}
\end{aligned}$$

Hence, by Observation C.17, the contribution of rows in $\mathcal{A}_j$ to the total squared loss is at least

$$\begin{aligned}
(1 - e^{-2^{-j-2}}) \cdot |\mathcal{A}_j| \cdot \frac{n^2}{2^{2(h_k-j)}} &= (1 - e^{-2^{-j-2}}) \cdot \frac{n^2}{2^{h_k-j-1}} \\
&= (1 - e^{-2^{-j-2}}) \cdot \frac{n^2}{2^{h_k-2}} \cdot 2^{j-1}
\end{aligned}$$

Thus, the contribution of rows with "large" squared norm, i.e., $\mathcal{A}_{>0}$, to the total squared loss is at least[7]

$$\frac{n^2}{2^{h_k-2}} \cdot \sum_{j=0}^{h_k} 2^{j-1} \cdot (1 - e^{-2^{-j-2}}) \ge 1.095 \cdot \frac{n^2}{2^{h_k-2}} \qquad \triangleright \text{for } h_k > 8 \qquad \square$$

Corollary C.19. *Let S_g be a CountSketch whose sparsity pattern is learned over a training set drawn from $\mathcal{A}_{sp}$ via the greedy approach. Let S_r be a CountSketch whose sparsity pattern is picked uniformly at random. Then, for an $n \times d$ matrix $A \sim \mathcal{A}_{zipf}$, for a sufficiently large value of k, the expected loss of the best* rank-k *approximation of A returned by S_r is worse than the approximation loss of the best* rank-k *approximation of A returned by S_g by at least a constant factor.*

Proof: The proof follows from Lemma C.18 and Corollary C.16. $\square$

Remark C.20. We have provided evidence that the greedy algorithm that examines the rows of A according to a non-increasing order of their norms (i.e., *greedy with non-increasing order*) results in a better rank-k solution compared to the CountSketch whose sparsity pattern is chosen

[7]The numerical calculation is computed by WolframAlpha.

at random. However, still other implementations of the greedy algorithm may result in a better solution compared to the greedy with non-increasing order. To give an example, in the following simple instance the greedy algorithm that check the rows of A in a random order (i.e., *greedy with random order*) achieves a rank-k solution whose cost is a constant factor better than the solution returned by the greedy with non-increasing order.

Let A be a matrix with four orthogonal rows u, u, v, w where $\|u\|_2 = 1$ and $\|v\|_2 = \|w\|_2 = 1+\varepsilon$ and suppose that the goal is to compute a rank-2 approximation of A. Note that in the greedy with non-decreasing order, v and w will be assigned to different bins and by a simple calculation we can show that the copies of u also will be assigned to different bins. Hence, the squared loss in the computed rank-2 solution is $1 + \frac{(1+\varepsilon)^2}{2+(1+\varepsilon)^2}$. However, the optimal solution will assign v and w to one bin and the two copies of u to the other bin which results in the squared loss of $(1+\varepsilon)^2$ which is a constant factor smaller than the solution returned by the greedy with non-increasing order for sufficiently small values of ε.

On the other hand, in the greedy algorithm with random order, with a constant probability $(\frac{1}{3}+\frac{1}{8})$, the computed solution is the same as the optimal solution. Otherwise, the greedy algorithm a with random order returns the same solution as the greedy algorithm with a non-increasing order. Hence, in expectation, the solution returned by the greedy with random order is better than the solution returned by the greedy algorithm with non-increasing order by a constant factor.

D Experiments - Appendix

We first note that we group k-means clustering into the low rank approximation (LRA) family of tasks because it can be considered as a constrained LRA problem (Section 2.3 in [CEM+15]).

D.1 Baselines

We comment on two of our baselines for the low-rank approximation family of tasks (LRA, k-means):

- **Exact SVD**: In the canonical, learning-free sketching setting (i.e., any matrix is equally probable), sketching using the top m singular vectors yields a $(1+\varepsilon)$-approximation for both LRA and k-means [CEM+15].
- **Column sampling**: In the canonical, learning-free sketching setting, sketching via column sampling yields a $(1+\varepsilon)$-approximation for k-means [CEM+15].

D.2 Evaluation metric

To evaluate the quality of a given sketch S, we first compute the task objective, $\mathcal{L}(S, A)$, over $\mathcal{A}_{\text{test}}$. We will denote this by $\mathcal{L}_{\text{test}}(S)$. As an example, for MRR:

$$\mathcal{L}_{\text{test}}(S) = \frac{1}{|(\mathcal{A},\mathcal{B})_{\text{test}}|} \sum_{(A_i,B_i)\in(\mathcal{A},\mathcal{B})_{\text{test}}} \left\| A_i (SA_i)^+ (SB_i) - B_i \right\|_F^2$$

Then, we compute the average optimal objective value over $\mathcal{A}_{\text{test}}$. We call this $\mathcal{L}^*$. Continuing the MRR example, we have:

$$\mathcal{L}^* = \frac{1}{|(\mathcal{A},\mathcal{B})_{\text{test}}|} \sum_{(A_i,B_i)\in(\mathcal{A},\mathcal{B})_{\text{test}}} \left\| A_i(A_i^\top A_i)^{-1}A_i^\top B_i - B_i \right\|_F^2$$

For MRR and LRA, it is easy to compute the optimal objective because we have expressions for the optimal solutions. For ℓ_p and Huber regression, we use stochastic (sub)-gradient descent (SGD) or iteratively reweighted least squares (IRLS) until convergence to iteratively find the optimal solution. For k-means clustering, we use an approximation algorithm such as k-means++ to approximate $\mathcal{L}^*$. Finally, our evaluation metric is $\Delta_S = \mathcal{L}_{\text{test}}(S) - \mathcal{L}^*$.

D.3 Multiple trials

We computed each value in our tables by averaging over three trials. For the tables in Section 8 that omitted standard deviations, we provide that information in Tables D.1 and D.2. For most experiments, the standard deviation is much smaller than the difference in approximation error between *random* and *learned (random pattern)*.

Table D.1: Test errors and standard deviations for $\ell_{1.5}$ regression

m, Sketch	Gas	Tunnel	Electric
20, random	16.100757 ± 1.878811	0.905101 ± 0.148539	139.079274 ± 7.094347
20, learned (random pattern)	**15.662183** ± 1.249631	**0.767999** ± 0.177185	**131.755758** ± 3.352831
30, random	14.955649 ± 0.207941	0.639754 ± 0.190440	57.653981 ± 0.801098
30, learned (random pattern)	**14.858939** ± 0.553371	**0.505537** ± 0.057456	**48.685349** ± 2.433104
40, random	14.329586 ± 0.162131	0.425036 ± 0.086589	48.733799 ± 1.633098
40, learned (random pattern)	**14.305626** ± 0.138766	**0.345803** ± 0.051074	**44.492352** ± 3.361606

Table D.2: Test errors and standard deviations for Huber regression

m, Sketch	Gas	Tunnel	Electric
20, random	0.058685 ± 0.029150	0.001115 ± 0.000043	0.288651 ± 0.017330
20, learned (random pattern)	**0.042292** ± 0.027936	**0.001091** ± 0.000054	**0.275271** ± 0.014699
30, random	0.045003 ± 0.023366	0.001279 ± 0.000291	0.108427 ± 0.019368
30, learned (random pattern)	**0.020599** ± 0.016536	**0.000987** ± 0.000079	**0.106119** ± 0.021819
40, random	0.043164 ± 0.008262	0.001006 ± 0.000057	0.073449 ± 0.011859
40, learned (random pattern)	**0.009170** ± 0.015103	**0.000918** ± 0.000047	**0.071874** ± 0.011770

D.4 Experimental parameters

For the tables in Section 8, we describe experimental parameters. First, we provide some general implementation details.

We implemented both the greedy (Algorithm 6) and stochastic gradient descent (Algorithm 2) algorithms in PyTorch. In the first case, PyTorch allowed us to harness GPUs to speed up computation on large matrices. We used several Nvidia GeForce GTX 1080 Ti machines. In the second case, PyTorch allowed us to effortlessly compute numerical gradients for each task's objective function. Specifically, PyTorch provides automatic differentiation, which is implemented by backpropagation through chained differentiable operators.

There are also two points of note in the greedy algorithm implementation. First, we noticed that for MRR and LRA, each iteration required computing the SVD for many rank-1 updates of the current S. Instead of computing the SVD from scratch for each of these variants, we first computed the SVD of S and then used fast rank-1 SVD updates [Bra06]. This greatly improved the runtime of Algorithm 6. Second, we decided to set $\mathcal{D}_w$ (the set of candidate row weights) to 10 samples in $[-2, 2]$ because we noticed most weights were in this range after running Algorithm 2.

Next, we introduce some abbreviations for our experimental parameters.

For Algorithm 2:

- bs: batch size, the number of training samples used in a given iteration.
- lr: learning rate.
- num_it: total number of gradient steps.
- $init\_method$: if not initializing with S produced by Algorithm 6, then this is the initialization method for the values of the non-zero entries. These can be Rademacher ($Unif(\pm 1)$) or standard Gaussian ($\mathcal{N}(0, 1)$).

For Algorithm 6:

- row_order: order that the rows of the data matrices are iterated over.
- bs: batch size, the number of training samples used in a given iteration.

Table 8.1: Test errors for MRR

Parameters for Algorithm 2: $bs = 20$, $lr = 10.0, 50.0, 30.0$ for gas, tunnel, and electric respectively, $num\_it = 1000, init\_method = Unif(\pm 1)$
Parameters for Algorithm 6: $row\_order = random$, $bs = 5$ for gas and $bs = 1$ for others

Table 8.2: Test errors for $\ell_{1.5}$ regression

Parameters for Algorithm 2: $init\_method = \mathcal{N}(0, 1)$ for all, other parameters in Table D.3

Table D.3: Experimental parameters for *learned (random pattern)* on $\ell_{1.5}$ regression

m	Gas	Tunnel	Electric
20	$bs = 32, lr = 1e-5, num\_it = 1000$	$bs = 64, lr = 5e-6, num\_it = 1000$	$bs = 128, lr = 1e-5, num\_it = 180$
30	$bs = 64, lr = 1e-5, num\_it = 200$	$bs = 64, lr = 5e-6, num\_it = 1000$	$bs = 64, lr = 1, num\_it = 90$
40	$bs = 32, lr = 1e-8, num\_it = 180$	$bs = 64, lr = 5e-6, num\_it = 1000$	$bs = 128, lr = 20, num\_it = 30$

Table 8.3: Test errors for Huber regression

Parameters for Algorithm 2: $num\_it = 1000$ unless otherwise stated, $init\_method = \mathcal{N}(0, 1)$ for all, other parameters in Table D.4

Table D.4: Experimental parameters for *learned (random pattern)* on Huber regression

m	Gas	Tunnel	Electric
20	$bs = 64, lr = 5e-03$	$bs = 64, lr = 0.5$	$bs = 64, lr = 1e-6, num\_it = 90$
30	$bs = 64, lr = 5e-03$	$bs = 64, lr = 50$	$bs = 128, lr = 1e-5, num\_it = 90$
40	$bs = 64, lr = 5e-03$	$bs = 128, lr = 50$	$bs = 128, lr = 1e-6, num\_it = 20$

Table 8.4: Test errors for LRA (using Algorithm 4 with four sketches)

For a given m, the dimensions of the four sketches were: $S \in \mathbb{R}^{m \times n}, R \in \mathbb{R}^{m \times d}, S_2 \in \mathbb{R}^{5m \times n}, R_2 \in \mathbb{R}^{5m \times d}$
Parameters for Algorithm 2: $bs = 5, lr = 1.0, 10.0$ for hyper and video respectively, $num\_it = 1000, init\_method = Unif(\pm 1)$

Table 8.5: Timing comparison for two approximate LRA algorithms

The algorithms are timed for speed on 1 input matrix run on an Nvidia GeForce GTX 1080 Ti GPU. The times are averaged over 10 trials.

Table 8.6: Test errors for LRA (using Algorithm 1 from [IVY19] with one sketch)

Parameters for Algorithm 2: $bs = 1, 5$ for *learned (random pattern)* and *learned (greedy pattern)* respectively, $lr = 1.0, 10.0$ for hyper and video respectively, $num\_it = 1000, init\_method = Unif(\pm 1)$
Parameters for Algorithm 6: $row\_order = random, bs = 1$

Table 8.7: Test errors for k-means clustering

We used the transposition of the LRA data sets and the transposition of the left-handed sketches trained for LRA. In other words, our sketched data was $A^\top S^\top$, with A from an LRA data set and S trained for LRA using Algorithm 1 from [IVY19] and the *learned (random pattern)* method.

Table 8.8: Test errors for *learned (greedy pattern)* with different row orders

Table values were computed for the LRA task with $k = 20, m = 20$.
We analyze row ordering performance in greater detail below.

Definition D.1 (Leverage score). *Let A_i be the i-th row of A. The leverage score of A_i is*

$$0 \leq \sigma_i = A_i^\top (A^\top A)^+ A_i \leq 1$$

A row's leverage score measures how important it is in composing $row(A)$. If a row has a large leverage score, that means it has components along the least well-represented row space directions and is therefore important.

We studied three arbitrary row orderings (*random, forwards, backwards*), the *decreasing row norm* ordering suggested by Theorems 7.1 and 7.2, and the decreasing leverage score ordering. We included this last ordering because it was reasonable to try prioritizing rows that were "unsubstitutable" in composing $row(A)$.

The performance of the *decreasing leverage score* ordering was average, perhaps because it is better to prioritize rows that compose important (i.e., high variance) directions of $row(A)$ than rows that compose *rare* directions (i.e., rows with high leverage scores). The *decreasing row norm* ordering sometimes performed very well and sometimes with a less pronounced effect. One conjecture is that it excels when the natural data distribution closely matches the spiked covariance or Zipfian models, as explained by our theory.

We note that the *forward* ordering performed best on the *synthetic spiked covariance* data set. This is because we constructed the data set by placing the heavy-normed rows first in every matrix. However, we can still observe that the *decreasing row norm* ordering is $\sim 20\%$ better than *random*, as suggested by Theorem 7.1.

Based on Table 8.8 and Tables 8.1 and 8.6 (where the *random* ordering was used exclusively), we conclude that while the best row ordering depends on the application, *random* performs well across various applications.

D.5 Running time

We examine the runtimes of our various sketching algorithms. In Table D.5, the times are obtained for the LRA task with $k = 30, m = 60$ on the logo data set. However, similar trends should be expected for other combinations of task, task parameters, and data sets.

We define the *inference* runtime as the time to apply the sketching algorithm. The *training* runtime is the time to train a sketch on $\mathcal{A}_{\text{train}}$ and only applies to learned sketches. Generally, the long *training* times are not problematic because training is only done once and can be completed offline. On the other hand, the *inference* runtime should be as fast as possible.

Note that *inference* was timed using 1 matrix from $\mathcal{A}_{\text{test}}$ on an Nvidia Geforce GTX 1080 Ti GPU. The values were averaged over 10 trials.

We observe that sparse sketches (such as the ones used in *learned (random pattern)* and *learned (greedy pattern)*) have much lower *inference* runtimes than the dense sketches of *exact SVD*.

Table D.5: Timing comparison of sketching algorithms for LRA (using Algorithm 1 from [IVY19])

	Time (sec)
random: inference	0.0114
exact SVD: inference	0.185
learned (random pattern): training	193 (3 min)
learned (random pattern): inference	0.0114
learned (greedy pattern): training	6300 (1 h 45 min)
learned (greedy pattern): inference	0.0114