Practical and Fast Embeddings for Large-Scale Regression

ABSTRACT

Today, optimization and regression tasks are a core requirement for modern data management systems handling machine learning workloads. Efficient algorithms that solve these problems, even approximately, are needed to work with large datasets. In this paper, we study techniques from matrix sketching, a dimensionality reduction technique, to address this need. Matrix sketching has recently been applied to solving a variety of convex constrained regression problems via paradigms such as "sketch-and-solve" (S&S) and "Iterative Hessian Sketching" (IHS). We study these two models, and show that the fast CountSketch and sparse Johnson-Lindenstrauss Transforms yield state-of-the-art accuracy guarantees under IHS, while dramatically improving the time cost. As a result, we obtain significantly faster algorithms for constrained regression, for both sparse and dense inputs. Our empirical results show that we can summarize data roughly 100x faster for sparse data, and, surprisingly, at least 10x faster on dense data! The consequence is that solutions accurate to within machine precision of the optimal solution can be found much faster than the previous state of the art.

CCS CONCEPTS

• **Theory of computation → Sketching and sampling**; **Numeric approximation algorithms**; Convex optimization; Quadratic programming.

KEYWORDS

large datasets, matrix sketching, regression

ACM Reference Format:

. 2019. Practical and Fast Embeddings for Large-Scale Regression. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Modern workloads have challenged the ability of traditional data management systems to adapt to a new set of demands. Workloads are more likely to be expressed in terms of building and applying machine learning models than as relational queries. Growing data sizes have prompted the adoption of approximate methods which complement exact solutions by offering reduced time or space requirements at the expense of tolerating some (small) error. Such settings can be modeled as solving large constrained optimization problems where data and constraints are modeled with matrices and vectors. Initial approaches were to use heuristics based on matrix factorization and random projection. More recently, the model of matrix sketching has emerged, where the projection of a data matrix with "sketch" matrices drawn from appropriate random families can give strong randomized approximation guarantees on the results. Performance is further enhanced when the sketch transformations can be applied quickly, due to enforced structure in the random sketches. In this work, we focus on problems of convex constrained regression, and show that a very sparse (and hence fast) sketching approach can outperform other methods, and be implemented within modern systems.

In our notation, matrices are written in upper case and vectors in lower case. A convex constrained least squares problem is specified by a sample-by-feature data matrix $A \in \mathbb{R}^{n\times d}$ with associated target vector $b \in \mathbb{R}^n$ and a set of convex constraints C. The error metrics will be expressed using the Euclidean norm $\|\cdot\|_2$ and the prediction (semi)norm $\|x\|_A = \frac{1}{\sqrt{n}}\|Ax\|_2$. The task is to find

$$x_{OPT} = \text{argmin}_{x\in C}\ \tfrac{1}{2}\|Ax - b\|_2^2. \tag{1}$$

In the large data setup we assume that $n \gg d$ so that solving (1) exactly is not possible with the resources available. A requirement to solve (1) exactly is computing A^TA in time proportional to nd^2 for system solvers. However, this dependence on n and d is sufficiently unfavourable that we must exploit some notion of approximation to solve (1) efficiently. Two competing methods of approximation through matrix sketching are the *sketch-and-solve* and *iterative Hessian sketching* models:

Sketch-and-solve (S&S): Approaches under the banner of "sketch-and-solve" output estimates found by solving

$$\hat{x} = \text{argmin}_{x\in C}\ \|S(Ax - b)\|_2^2 \tag{2}$$

by sampling a random linear transformation $S \in \mathbb{R}^{m \times n}$ from a sufficiently well-behaved distribution of matrices with $m \ll n$. Now that m is much smaller than n, the $m \times d$ dimensional problem is small enough that it can be solved exactly. In fact, for appropriately chosen S, one can show that the *cost* of a regression problem can be aproximated accurately in the sense that $\|A\hat{x} - b\|^2 \leq (1+\varepsilon)\|Ax_{OPT} - b\|^2$ where ε is an accuracy parameter and the result holds with high constant probability. It is assumed that the projection dimension m is chosen sufficiently small relative to n that the smaller problem is efficiently solvable and that the main computational bottleneck is the time it takes to compute the summary SA.

Iterative Hessian Sketching (IHS): This approach exploits the quadratic program formulation of (1) and uses random projections to accelerate expensive computations in the problem setup. In contrast to the S&S approach, the aim here is to follow an iterative scheme which gradually refines the estimate in order to descend to the true solution of the problem, via steps defined by (3).

$$x^{t+1} = \underset{x \in C}{\operatorname{argmin}} \frac{1}{2}\|S^{t+1}A(x - x^t)\|_2^2 - \langle A^T(b - Ax^t), x - x^t \rangle \quad (3)$$

The benefit of this approach is that (3) is a special instance of a quadratic program where the norm term is computed by obtaining a Hessian matrix. To solve this program without sketching (i.e. set $S = I$), one needs to compute the exact Hessian matrix A^TA which takes $O(nd^2)$. However, we will approximate A^TA by the quadratic form $(SA)^TSA$ instead and any further access to A is for matrix-vector products rather than a full matrix-matrix product. Using a projection dimension of m rows, we compute $\|SAx\|_2^2 = x^TA^TS^TSAx$ (for variable x) in time $O(T_{\text{sketch}} + md^2)$, using an approximate Hessian which is sufficiently-well concentrated around its mean to ensure that the iterative scheme enjoys convergence to the optimal solution of the problem. This is a huge computational saving when we compute the quadratic form since SA has $m \ll n$ rows.

The per-iteration time costs comprise the time to sketch the data, T_{sketch}, the time to construct the QP, T_{QP} and the time to solve the QP, T_{solve}. In our setting, we will always bound T_{solve} by $O(d^3)$ for the cost of quadratic programming. For a projection dimension m then $T_{\text{QP}} = O(md^2 + nd)$ to construct the approximate Hessian and for all inner product terms. We may have that $T_{\text{sketch}} = O(mnd)$ for a fully dense sketch matrix, such as the Gaussian sketches described in Section 2, resulting in no substantial computational gain. Instead, we consider sketching techniques that can be computed more quickly. Our main technical and empirical results show that these choices can be highly advantageous in real systems.

1.1 Related Work

Prompted by the increasing size of data, there is need to design algorithms and systems which can scale up and meet new demands. The data management community has focused on this growing need to address scalability by studying data analysis problems in both distributed and streaming models of computation. Practical machine learning and database systems have converged within deployed architectures, for instance via distributed platforms such as Apache Spark [3], Mahout [2], and also streaming platforms such as Kafka [1]. The algorithmic theory for fundamental analysis tasks (such as ℓ_p-norm, with $p > 0$ or $p = 0$ for distinct elements) approximation has been studied in the distributed model [33] as well as in streaming models (see, for instance [16, 17, 19]). Correspondingly, optimization routines have also been studied in both streaming and distributed models: least squares regression on a data stream was explored in [7] while the *Hogwild!* algorithm proposed in [25] estimates solutions to optimization problems in a distributed model. In both these examples, sparsity in the data is an important consideration. For example, by *assuming sparsity in the data* in Hogwild, different workers in the network can overwrite intermediate estimates, thus avoiding communication bottlenecks.

A popular technique is to use sketches to maintain estimates of the gradient values to enable fast updates to the weight vectors. This was explored in both the streaming setting [28] and the distributed setting [14]. Outside of distributed or streaming models, there is much effort given to batch gradient descent when the entire dataset is available in one database, such as [18, 27].

Our work sits at the intersection of much of the above work from both the database and machine learning community/ We focus on implementing time-efficient algorithms that exploit sparsity to solve constrained optimization problems (i.e., regression). Within this family of convex constrained least squares problems are popular data analysis tools such as ordinary least squares (OLS, $C = \mathbb{R}^d$), and penalised regression: $C = \{x : \|x\|_p \leq t, p = 1, 2\}$ as well as as Elastic Net and SVM. For OLS or LASSO (penalised regression with $p = 1$), the time complexity of solving the optimisation problem is $O(nd^2)$. We now briefly survey works which have studied such regression problems under S&S and IHS perspectives.

Sketch-and-solve (S&S): When a sketch of a matrix is a *subspace embedding* (defined formally in Section 2), it approximately preserves norms of all directions after projection. This allows the cost of a regression problem to be approximated up to $(1 \pm \varepsilon)$ *relative error* [32]. Sarlós used *Fast Johnson Lindenstrauss Transforms* (FJLT) to construct subspace embeddings to find the first $o(nd^2)$ algorithm for solving unconstrained regression [26]. A similar idea was employed by

Clarkson and Woodruff [8] to generate a subspace embedding in time proportional to the number of nonzeros in the matrix (denoted nnz(A) throughout). Such ideas have been extended to ridge regression in [5] as well as other forms of constrained regression, surveyed in [32].

A particular strength of S&S is that only one pass of the data is needed and as such it is attractive for extremely large datasets which are never recorded in full. However, a drawback of sketch-and-solve is that although one can accurately preserve the objective value of the original problem, the projection onto a random subspace yields suboptimal estimators of the *solution vector* to the original problem. Despite its strengths, sketch-and-solve fares worse when new examples are added, which is a major concern when data is evolving. A partial guarantee was shown by Price *et al.* [24] with an ℓ_∞ guarantee on the distance between the approximate estimator and the optimal solution vector. This work also showed pathological instances for the CountSketch, which is the method of primary interest in our work. A particular contribution of our work is to rehabilitate CountSketch for approximate regression, within the IHS framework. Recent work of Dahiya *et al.* [10] has motivated the use of CountSketch for several data analysis problems. That work showed theoretical and empirical results for using CountSketch within the S&S paradigm to solve ordinary least squares, robust regression, and other linear algebra problems. The central point of departure for our work is the adoption of the stronger IHS model, which allows convex constraints as well as solution approximation (the latter being a key drawback of the sketch-and-solve model), and comparing the CountSketch to general sparse embeddings in this framework.

Iterative Hessian Sketch (IHS): The IHS approach proposed by Pilanci and Wainwright [20] takes the iterative approach of (3) instead of directly solving $\min \|S(Ax-b)\|^2$. The benefit of this is that after sufficiently many steps, the output approximation, $\hat{x}$, has small error from the true optimal solution. This idea was later refined to sketch both large (n) and high dimensional problems (d) through an iterative primal-dual approach in [31] but applied specifically to the case of high-dimensional ridge regression. The IHS method has been extended to handle a wider class of problems, those whose objective function is twice-differentiable and convex [21].

Between the two models is the sketched gradient descent of Clarkson and Woodruff [9]. Here, the sketch is applied once to obtain SA, Sb which is used to solve the regression problem to machine precision in the reduced space using gradient descent. At a high level, the idea is similar to IHS, albeit this result obtains a *cost approximation* as opposed to the *solution approximation* of IHS.

1.2 Contributions

Our primary focus is to understand the behavior of sparse embedding to solve constrained regression problems, in terms of time cost and embedding dimension. While the potential use of sparse embeddings is mentioned in the conclusion of [21], their use has not been studied in the IHS model (or its later variations).

- First, we establish a lower bound on the sketch dimension required for solution approximation with the CountSketch. This shows that in the (standard) Gaussian design setting using the S&S model, only suboptimal estimators of the weight vector can be obtained using small space. Thus, using the CountSketch within IHS is well-motivated.
- Second, we provide the necessary theoretical justification to allow the use of CountSketch and sparse embeddings within the IHS framework. Hence, iterations within the IHS framework can be performed in *input sparsity* time, which is a huge saving for sparse data.
- Next, we give comprehensive baseline experiments to understand how sparse embeddings perform in comparison to the dense sketches. In line with the theory, CountSketch is practically a much faster summary method as it can be computed while 'streaming' over the dataset. In addition, we show that the error behavior of CountSketch on real data is better than the theory predicts.
- Finally, we *compare various sketching methods* empirically on instances of constrained regression. We see that CountSketch performs better than alternate sketches across a range of synthetic and real datasets. Our main experimental finding is that noisy estimates with sparse embeddings can be found much quicker than accurate estimates with dense sketches, yielding significant speedups overall.

Our work then gives much cause for optimism about the practicality of adopting random sketching in systems for large scale data analysis. Our experimental contribution culminates in showing the practical benefits of using CountSketch: often IHS with CountSketch converges to machine precision before the competing methods have completed even two iteration steps.

Outline. Background material is given in Section 2 and theoretical discussions in Section 3. Sections 4 and 5 are then used to examine the behaviour of the competing methods. We provide rigorous baseline experiments to justify why we use the CountSketch at the same projection dimension as competing methods; these baselines vindicate our claim that CountSketch performs better in practice than one should expect from the theory (regarding space and error). Section

6 highlights the benefits of our approach by showing how an approximation to an optimal LASSO solution can be found much faster using sparse embeddings in the IHS.

2 PRELIMINARIES

First we introduce notation and terminology for the required theory and describe the random projections that define different sketching methods. Let $A \in \mathbb{R}^{n\times d}$ be the input data matrix and let m be the desired projection dimension.

Gaussian sketch: Sample a matrix G whose entries are iid normal, $G_{ij} \sim N(0,1)$, and define the sketch matrix S by scaling $S = G/\sqrt{m}$.

Subsampled Randomized Hadamard Transform (SRHT) [4]: Define $S = PHD$ based on three simply structured matrices. D is a diagonal matrix with $D_{ii} \in \{\pm 1\}$ each with probability $1/2$ independently; H is the recursively defined Hadamard Transform; and P is a matrix which samples rows uniformly at random.

CountSketch [32]: Initialise $S = \mathbf{0}_{m,n}$ and for every column i of S choose a row $h(i)$ uniformly at random. Set $S_{h(i),i}$ to either $+1$ or -1 with equal probability.

Sparse Johnson-Lindenstrauss Transform (SJLT) [15]: The sparse embedding S with column sparsity (number of nonzeros per column) parameter s is constructed by row-wise concatenating s independent CountSketch transforms, each of dimension $m/s \times n$. Hence, a sparse embedding with $s = 1$ is a standard CountSketch.

Both CountSketch and SJLT will be collectively referred to as *sparse embeddings*.

2.1 Complexities of Random Projections

The key property we need is given in the following definition.

Definition 2.1. A matrix $S \in \mathbb{R}^{m\times n}$ is a $(1 \pm \varepsilon)$-*subspace embedding* for the column space of a matrix $A \in \mathbb{R}^{n\times d}$ if for all vectors $x \in \mathbb{R}^d$, $\|SAx\|_2^2 \in [(1-\varepsilon)\|Ax\|_2^2, (1+\varepsilon)\|Ax\|_2^2)]$.

Each family of random matrices defined above provide subspace embeddings with at least constant probability for m large enough, summarized as follows:

Theorem 2.2 ([32])**.** Let $A \in \mathbb{R}^{n\times d}$ have full column rank. Let $S \in \mathbb{R}^{m\times n}$ be sampled from one of the Gaussian, SRHT, or CountSketch distributions. Then to achieve the subspace embedding property with probability $1-\delta$ we require $m = O(\varepsilon^{-2}(d+\log(1/\delta)))$ for the Gaussian and SJLT sketches; $m = \Omega(\varepsilon^{-2}(\log d)(\sqrt{d}+\sqrt{n})^2)$ for the SRHT; and $m = O(d^2/(\delta\varepsilon^2))$ for the CountSketch.

A summary of the sketching methods is given in Table 1. While the ε dependence of projection dimension m for all methods is the same (ε^{-2}), the behavior as a function of d is

Table 1: Sketch complexity for subspace embeddings. In SJLT, s is the number of nonzeros per column.

Sketch Method	Embedding Dimension (m)	Projection Time
Gaussian	$O\left(d\varepsilon^{-2}\log\frac{1}{\delta}\right)$	$O(nd^2)$
SRHT	$O\left(d\varepsilon^{-2}\log\frac{1}{\delta}\right)$	$O(nd\log n)$
CountSketch	$O\left(d^2/\delta\varepsilon^2\right)$	$O(\text{nnz}(A))$
SJLT	$O\left(d\varepsilon^{-2}\log\frac{1}{\delta}\right)$	$O(s\cdot\text{nnz}(A))$

variable. Each of Gaussian, SJLT and SRHT require m to be (at worst) d poly $\log d$, while the CountSketch, although faster to apply, depends on d^2. This suggests that CountSketch would not be preferred as the dimension d increases. Moreover, the failure probability of CountSketch depends *linearly* upon δ whereas both Gaussian and SRHT depend on $\log 1/\delta$. If this behavior is observed in practice, it could make CountSketch inferior to other sketches, since we typically require δ to be very small. In our subsequent empirical evaluation, we evaluate the impact of these two parameters, and show that we nevertheless obtain very satisfactory error behavior, and extremely high speed.

Time Complexity of sketching methods: Each of the described random projections defines a linear map from $\mathbb{R}^n \to \mathbb{R}^m$ which naively would take $O(mnd)$. Despite this, only the Gaussian sketch incurs the dense matrix multiplication time cost. Implementing SRHT exploits the fast Hadamard transform which takes $O(nd\log d)$[1] time as it is defined recursively, while applying P and D take time $O(n)$. The CountSketch can be applied by streaming through the matrix A: upon observing a (non-zero) entry A_{ij}, the value of a hash bucket defined by the function h is then updated with either $\pm A_{h(i),j}$. Thus the time to compute a CountSketch transform is proportional to $\text{nnz}(A)$ and $s\cdot\text{nnz}(A)$ for the general sparse embedding.

3 TECHNICAL RESULTS

3.1 S&S Suboptimality with CountSketch

We assume the standard Gaussian design for our problems, which states that $y = Ax^* + \omega$ (note though that the results from S&S hold with no assumptions on the noise vector ω). Here, the data A is fixed and there exists an unknown ground truth vector x^* belonging to some compact $C_0 \subseteq C$, while the error vector ω has entries drawn by $\omega_i \sim N(0,\sigma^2)$. In order to compare the output of an algorithm, $\hat{x}$, to that of the ideal estimator, x_{OPT}, Pilanci and Wainwright showed that the expected solution error could be bounded above as a function of the input size and from below as a function of the sketch size (Theorem 3.1). Unfortunately, a similar

[1] See [32] for details on the improvement from $O(nd\log n)$ to $O(nd\log d)$

upper bound on the prediction error (Proposition 3.2 below) requires the sketch dimension m to grow roughly to the same order as n (i.e. no reduction in size) in order to provide commensurate error to the 'true' weights as the optimal estimator. In addition, it was shown that any estimator x' which observes the pair (SA, Sb) is provably suboptimal in the sense outlined in Theorem 3.1. The key condition is that a sketching matrix has the following property (in spectral norm):

$$\|\mathbb{E}[S^T(SS^T)^{-1}S]\|_2 \leq \eta \frac{m}{n} \tag{4}$$

Theorem 3.1 ([20]). Let $S \in \mathbb{R}^{m\times n}$ be a sketching matrix which satisfies (4). Any estimator based on observing (SA, Sb) has solution error bounded as:

$$\sup_{x^*\in C_0} \mathbb{E}_{S,\omega}[\|x^* - x'\|_A^2] = \Omega\left(\frac{\sigma^2}{\eta \min(m,n)}\right). \tag{5}$$

To see the issue of S&S suboptimality we also need:

Proposition 3.2 ([20]). For any set C containing x^*, the constrained least-squares estimate x_{OPT} has prediction error upper bounded as:

$$\mathbb{E}\|x_{OPT} - x^*\|_A^2 \leq c\left(\varepsilon_n(\bar{\mathcal{K}}) + \frac{\sigma^2}{n}\right). \tag{6}$$

For example, the unconstrained case has the prediction error is $O(\sigma^2 d/n)$.

For random projections that we study, previous work showed Equation (4) holds for the SRHT and Gaussian transforms [20]. Our key result here is in establishing Equation (4) for the CountSketch. This result for the CountSketch is summarized as follows in Theorem 3.3.

Theorem 3.3. If S be an $m \times n$ CountSketch matrix with no all-zero rows, the spectral bound (4) is met with $\eta = 1$.

The proof is given in Appendix A and the main idea is to analyse the CountSketch as a random linear transform through the behavior of the hash functions. The random structure of the hash functions used to construct S allows us to understand the matrix products in Equation (4) and derive the spectral bound. Note that the restriction of no zero rows is a convenient simplification, since the proofs involves the inverse of a matrix, which is not guaranteed to exist if the sketch has zero rows. However, this is not a limitation, since we can simply consider the sketch with zero rows removed (corresponding to buckets which have no input elements mapped to them).

In summary, CountSketch suffers the same deficiencies as other sketches when used in S&S, which is perhaps unsurprising. However, this further motivates its adoption in the IHS framework.

3.2 IHS with Sparse Embeddings

Throughout, we consider instances of overconstrained regression (1) given in the form of a data matrix $A \in \mathbb{R}^{n\times d}$, target vector $b \in \mathbb{R}^n$ with $n \gg d$ and convex constraints C. Our aim is to provide approximation guarantees on the solutions found for this system. These may be either *cost* or *solution* approximations. Let $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ for input parameters (A, b) and let x_{OPT} denote the optimal (constrained) solution of our instance.

Definition 3.4. An algorithm which outputs $\hat{x}$ is a $(1+\varepsilon)$ *cost approximation* if $f(\hat{x}) \leq (1+\varepsilon)f(x_{OPT})$.

Definition 3.5. An algorithm which returns $\hat{x}$ such that $\|x_{OPT} - \hat{x}\|_A \leq \varepsilon\|x_{OPT}\|_A$ is referred to as a *ε-solution approximation* algorithm, where $\|x\|_A = \frac{1}{\sqrt{n}}\|Ax\|_2$.

Recall that the S&S approach directly solves a sketched instance of the problem (as in Equation (2)) to provide a cost approximation as in Definition 3.4. To avoid the solution approximation suboptimality of the S&S approach, the IHS approach performs a sequence of sketching steps via (3).

Time Costs. The time taken for each approach is similar, and is determined by the combination of the sketch time T_{sketch} and the time to solve a smaller problem in the 'sketch space', T_{solve}. We typically project data to a $\tilde{O}(d) \times d$ size matrix, so T_{solve} is $\tilde{O}(d^3)$[2]. For the IHS method, we typically require $O(\log 1/\varepsilon)$ iterations to reach convergence, whereas sketch-and-solve is a 'one-shot' algorithm.

We next outline the technical arguments needed to use CountSketch and SJLT in IHS.

3.2.1 *Subgaussianity of CountSketch.*

Lemma 3.6. Let S be a CountSketch matrix. Let N_i denote the number of nonzeros in row S_i. Then SS^T is a diagonal matrix with $(SS^T)_{ii} = N_i$ and hence distinct rows of S are orthogonal.

Proof. The entries of matrix SS^T are given by the inner products between pairs of rows of S. Consequently, we consider the inner product $\langle S_i, S_j\rangle$. By construction S has exactly one non-zero entry in each column. Hence for distinct rows $i \neq j$, $\langle S_i, S_j\rangle = 0$. Meanwhile, the diagonal entries are given by $\|S_i\|_2^2 = \sum_{j=1}^n S_{ij}^2$, i.e. we simply count the number of non-zero entries in row S_i, which is N_i. □

Lemma 3.7. $\mathbb{E}[SS^T] = \frac{n}{m} I_m$

Proof. Continuing the previous analysis, we have that $(SS^T)_{ij} = \langle S_i, S_j\rangle$, the inner product between rows of S. Taking the expectation, $\mathbb{E}[S_i \cdot S_j] = \sum_{k=1}^n \mathbb{E}[S_{ik}S_{jk}]$. By Lemma 3.6, we know that for $i \neq j$ then $\langle S_i, S_j\rangle = 0$ and hence

[2] If we take the $O(d^2)$ projections for CountSketch to give a subspace embedding, then T_{solve} is $\tilde{O}(d^4)$.

$\mathbb{E}[S_i \cdot S_j] = 0$. Otherwise, $i = j$ and we have a sum of n random entries. With probability $\frac{1}{m}$, $S_{ik}^2 = 1$, coming from the two events $S_{ik} = -1$ with probability $\frac{1}{2m}$, and $S_{ik} = 1$, also with probability $\frac{1}{2m}$. Then by linearity of expectation we have $\mathbb{E}[S_i \cdot S_i] = \sum_{k=1}^{n} \mathbb{E}S_{ik}^2 = n/m$. □

Lemma 3.8. The covariance matrix is an identity, $\mathbb{E}[S^T S] = I_{n \times n}$

Proof. Observe that $\mathbb{E}[S^T S]_{ij} = \langle S_i^T, S^j \rangle = \langle S^i, S^j \rangle$ so now we consider dot products betwen columns of S. Recall that each column S^i is a basis vector with unit norm. Hence for $i = j$, $\langle S^i, S^i \rangle = 1$. For $i \neq j$, the inner product is only non-zero if S^i and S^j have their non-zero entry in the same location, k. The inner-product is 1 when S_{ki} and S_{kj} have the same sign, and -1 when they have opposite signs. The probability of these two cases are equal, so the result is zero in expectation. In summary then, $\mathbb{E}[S^T S]_{ij} = 1$ if $i = j$ and 0 otherwise which is exactly the $n \times n$ identity matrix. □

Definition 3.9. A zero-mean random vector $s \in \mathbb{R}^n$ is sub-Gaussian if for any $u \in \mathbb{R}^n$ we have $\forall \varepsilon > 0, \mathbb{P}[\langle s, u \rangle \geq \varepsilon \|u\|_2] \leq \exp(-\varepsilon^2/2)$.

Lemma 3.10 (Rows of CountSketch are sub-Gaussian)**.** Let S be an $m \times n$ random matrix sampled according to the CountSketch construction. Then any row S_i of S is 1-sub-Gaussian.

Proof. Fix a row S_i of S and let $X = \langle S_i, u \rangle$. If either S_i or u is a zero vector then the inequality in the sub-Gaussian definition is trivially met, so assume that this is not the case. We need Bernstein's Inequality: when X is a sum of n random variables $X_1, \ldots, X_n$ and $|X_j| \leq M$ for all j then for any $t > 0$

$$\mathbb{P}(X > t) \leq \exp\left(- \frac{t^2/2}{\sum_j \mathbb{E}X_j^2 + Mt/3} \right). \quad (7)$$

Now consider $X = \sum_{j=1}^{n} S_{ij} u_j$, which is a sum of zero-mean random variables. We have $|X_j| = |S_{ij} u_j| \leq \|u\|_2$ for every j and $\mathbb{E}X_j^2 = u_j^2/m$. Taking $t = \varepsilon \|u\|_2$ in Equation (7) and cancelling $\|u\|_2^2$ terms gives $\mathbb{P}(X > \varepsilon \|u\|_2) \leq \exp\left(-\frac{\varepsilon^2/2}{1/m + \varepsilon/3}\right)$. The RHS is at most $\exp(-\varepsilon^2/2)$ whenever $1/m + \varepsilon/3 \leq 1$, which bounds $\varepsilon \leq 3 - 3/m$. Imposing $m > 1$, this is satisfied for all $\varepsilon \in (0, 1)$. □

3.2.2 Instantiating the Iterative Hessian Sketch. First we need that the sketches are zero mean and have identity covariance $\mathbb{E}[S^T S] = I_n$ which is shown in Lemma 3.8. Note that for SJLT with $s > 1$ we need to normalise by the number of nonzeros to ensure identity covariance. Two further properties are required for the error bounds of the IHS. These are given in Equation (9). However, to define these properties, we first introduce Definition 3.11, after which we can present the Theorem.

Definition 3.11. The *tangent cone* is the following set:

$$K = \{v \in \mathbb{R}^d : v = tA(x - x_{OPT}),\ t \geq 0, x \in C\} \quad (8)$$

We note that the residual error vector for an approximation $\hat{x}$ belongs to this set as $u = A(\hat{x} - x_{OPT})$. Let $X = K \cap \mathcal{S}^{n-1}$ where $\mathcal{S}^{n-1}$ is the set of n-dimensional vectors which have unit Euclidean norm. The quantities we must analyze are:

$$Z_1 = \inf_{u \in X} \|Su\|_2^2 \quad \text{and} \quad Z_2 = \sup_{u \in X} |u^T S^T S v - u^T v|. \quad (9)$$

Here, v denotes a *fixed* unit-norm n-dimensional vector. We assume S is a subspace embedding for the column space of A.

Theorem 3.12. Let $A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n$ and C be a set of convex constraints which define a *convex constrained least squares problem* whose solution is x^*. Fix an initial error tolerance $\varepsilon_0 \in [0, 1/2)$. Conditioned on the event that $Z_1 \geq 1 - \varepsilon_0$ and $Z_2 \leq \varepsilon_0/2$ for every iteration $1 \leq i \leq N$, then the IHS method returns an estimate with $\|\hat{x} - x^*\|_A \leq \varepsilon_0^N \|x^*\|_A$. Consequently, an error of $\varepsilon_0^N = \varepsilon$ can be achieved by choosing $N = \Theta(\log(1/\varepsilon))$. *In addition, all iterations can be performed in time proportional to* $O(nnz(A))$.

Remark 3.13. A lower bound on the column sparsity, namely $s = \Omega(d/\varepsilon)$ nonzeros per column, is needed to achieve a Johnson-Lindesntrauss Transform (JLT) [15]. It was also shown that if $s = \Omega(1/\varepsilon)$ then an SJLT will return a JLT. Hence, if S is an SJLT subspace embedding with s large enough, then the requirement on Z_2 is immediately met. However, for $s = 1$, the CountSketch does not in general provide a full JLT due to the $\Omega(d/\varepsilon)$ lower bound. This result prevents the CountSketch, which has only a single nonzero in every column, from being a JLT, unless the data satisfies some strict requirements [4].

Proof. We focus separately on the cases when S is an SJLT or a CountSketch. Observe that when S is a subspace embedding for the column space of A, we have for $u \in K \subseteq \text{col}(A)$ that $\|Su\|_2^2 = (1 \pm \varepsilon)\|u\|_2^2 = (1 \pm \varepsilon)$. Therefore, $Z_1 \geq 1 - \varepsilon$. Recalling Theorem 2.2, we see that this is achieved for the SJLT with $m = \tilde{O}(d \log d)$ provided the column sparsity s is sufficiently greater than 1. In contrast, for $s = 1$, CountSketch has $m = \tilde{O}(d^2)$ (here, $\tilde{O}$ suppresses the dependency on ε).

Next we focus on achieving $Z_2 \leq \varepsilon/2$ for both sketches. If $s = \Omega(1/\varepsilon)$ then the SJLT immediately satisfies (9) by virtue of being a JLT. Note that this matches the lower bound on the column sparsity as stated in Remark 3.13. In light of the column sparsity lower bound, we appeal to a different argument in order to use CountSketch within the IHS.

We invoke the approximate matrix product with sketching matrices for when $S \in \mathbb{R}^{m \times n}$ is a subspace embedding for col(A) (Theorem 13 of [32] restated from [15]). Define the matrices $U, V \in \mathbb{R}^{n \times d}$ whose first rows are u and v, respectively, followed by $n - 1$ rows of zeros. Now apply

the CountSketch S to each of U and V which will be zero except on the first row. Hence, the product U^TS^TSV contains $\langle Su, Sv\rangle$ at $(U^TS^TSV)_{1,1}$ and is otherwise zero; likewise U^TV contains only $\langle u, v\rangle$ at $(U^TV)_{1,1}$. The approximate matrix product result gives $\|U^TS^TSV - U^TV\|_F \leq 3\varepsilon\|U\|_F\|V\|_F$ for $\varepsilon \in (0, 1/2)$; from which desired property follows after rescaling ε by a constant. The definitions of U, V, u, v mean $\|U\|_F = 1, \|V\|_F = 1$ and hence the error difference $U^TS^TSV - U^TV$ has exactly one element at $(i, j) = (1, 1)$. This is exactly $|u^TS^TSv - u^Tv|$ and hence $Z_2 \leq \varepsilon$ after rescaling. Coupled with the fact that the rows of a CountSketch matrix S are sub-Gaussian, we are now free to use the CountSketch within the IHS framework.

Iteration Time Costs. The iterations require computing a subspace embedding at every step. The time cost to do this with sparse embeddings is: $O(\text{snnz}(A))$ for sketching A. Additionally we require (i) $O(md^2)$ to compute the approximate Hessian and (ii) $O(\text{nnz}(A))$ for the inner product terms in order to construct the intermediate quadratic program as in Equation (3). Solving the QP takes $\text{poly}(d) = \tilde{O}(d^3)$ when $m = \tilde{O}(d)$ for the SJLT (the $\tilde{O}$ suppresses small log factors). For CountSketch, $m = \tilde{O}(d^2)$, we require $\text{poly}(d) = \tilde{O}(d^4)$. Overall, this is $O(\text{nnz}(A) + d^3)$ for SJLT and $O(\text{nnz}(A) + d^4)$ for CountSketch. □

To summarize, we have shown that every iteration of the IHS can be completed in time proportional to the number of nonzeros in the data, with some (small) additional overhead to solve the QP at that time. Although the small polynomial factor for solving is larger for the CountSketch than the SJLT, in practice, this increased time cost is not observed on the datasets we test. In comparison to previous state of the art, the per-step time cost using the SRHT will be $O(nd \log d + d^3)$ and $O(nd^2 + d^3)$ for the Gaussian random projection.

4 EXPERIMENTAL CONFIGURATION AND CALIBRATION

4.1 Implementation Details

All experiments are performed in the Anaconda distribution of Python. We use a fast library for computing the Hadamard Transform[3] and the Numba library[4] to accelerate the Python code. Quadratic programs were solved using the CVXOPT library [29]. We test our algorithms on real and synthetic data, summarised in Table 2. Real datasets with $n \gg d$ were taken from the UCI Machine learning repository [12], the OpenML repository [30], and the Florida Sparse Matrix Collection [11]. Density was measured by taking $\text{nnz}(A)/(nd)$ and the data we test ranges from 1%–100% dense. To not disadvantage SRHT, we restrict the input matrix height to be a power of 2. We performed minimal preprocessing to the data as found in the repositories, apart from correcting for missing data values by removing the corresponding sample. For the Specular data, 50 of 1600 columns subsampled uniformly at random without replacement. The download and preparation steps are automated for reproducibility.

[3] https://bitbucket.org/vegarant/fastwht

[4] https://numba.pydata.org/

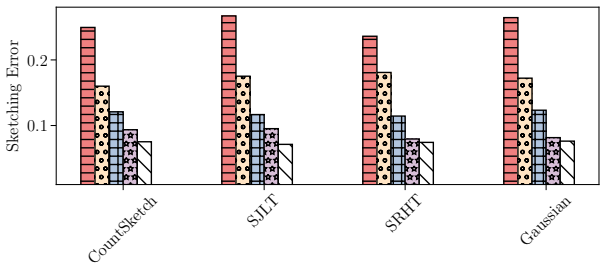
Table 2: Summary of datasets.

Dataset	Size	Density	Source
Abtaha	(37932, 330)	1%	[11]
Albert	(425240, 78)	84%	[30]
Aloi	(10800, 128)	24%	[30]
APSFailure	(76000, 170)	60%	[30]
CoverType	(581012, 54)	22%	[30]
FARS	(100968, 29)	80%	[30]
KDDCup99	(494020, 41)	100%	[30]
Specular	(477976, 50)	1%	[30]
W4A	(7366, 300)	4%	[11]
W6A	(17188, 300)	4%	[23]
W8A	(49749, 300)	4%	[23]
YearPredictionsMSD	(515344, 90)	100%	[12]

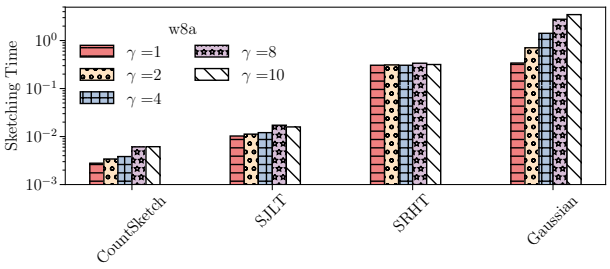
4.2 Subspace Embedding Calibration

We begin with a series of rigorous baseline experiments on real datasets. Calibration experiments were conducted on each of the datasets in Table 2 which have varying aspect ratios (d/n) and densities. We test each of the sketch methods at projection dimension $m = \gamma d$ for $\gamma \in \{1, 2, 4, 8, 10\}$ over 10 trials and report the mean values of the following recorded measurements. For the SJLT we use a column sparsity of $s = 4$ for comparison (this choice is studied in more detail below in Section 4.3). We measure the *empirical error* of each sketch in Frobenius and spectral norms: $\varepsilon = \|(SA)^TSA - A^TA\|/\|A^TA\|$. This condition is equivalent to testing the quality of a subspace embedding by writing Definition 2.1 in the matrix-vector product form. We observed that the behavior in both norms was similar up to rescaling so we only present the Frobenius error. Recall that from Theorem 2.2, CountSketch requires $m = \tilde{O}(d^2)$ compared to $m = \tilde{O}(d)$ for all other methods. Thus for a fixed $m = \gamma d$ we expect the empirical CountSketch error to be worse compared to the other sketching methods. We measure the *sketching time* to understand when competing methods are faster than others: the theory predicts that sparse embeddings with s nonzeros per column need $O(s \cdot \text{nnz}(A))$, so we expect these to be faster for sparse data, but not necessarily all data.

An example of these results is illustrated for the w8a dataset. The error across the competing methods is compared in Figure 1a and shows that on this data there is very comparable performance across all sketching methods for



(a) Empirical Sketching Error for w8a dataset



(b) Sketching time

Figure 1: Empirical measurements for w8a dataset (Legend refers to both plots)

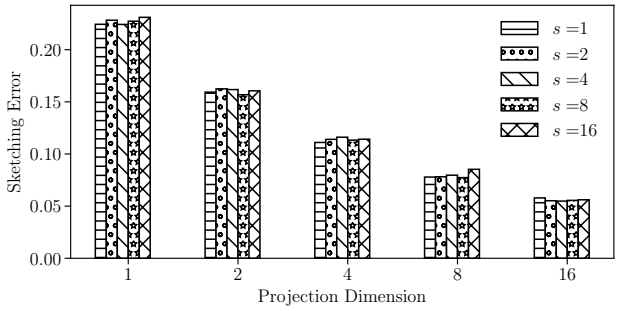


Figure 2: Empirical sketching error for SJLT compared to projection dimension factor γ and column sparsity (s) on w4a

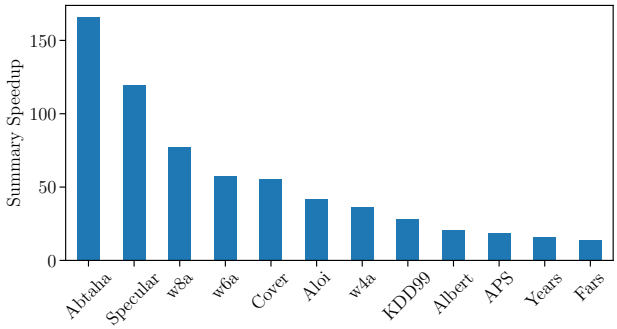


Figure 3: Summary speedup on all datasets

each given choice of γ. Interestingly, we observe that the CountSketch results are in line with the other methods, despite the more cautious theoretical guarantees. The corresponding time costs are given in Figure 1b, on log-scale. Here, there is a mild increase in the sketch time for the CountSketch as we increase γ. The SJLT (with $s = 4$) takes longer to apply but does slightly better than the linear factor of four times the time for Countsketch. Both sparse methods are between factors of 10–150 times faster than the SRHT. Although the Gaussian looks competitive with the SRHT on this dataset, as n increases the difference can become quite substantial, often approaching 2 orders of magnitude on the datasets that we test.

4.3 Sparse Embedding Calibration

To determine how best to choose the column sparsity parameter s for the IHS experiments we need to first understand how varying s affects the quality of the sketches. Behavior in both baseline experiments was fairly uniform across all datasets so for ease of illustration and space we provide only the plots for the w4a dataset. For this experiment we replicate the setting of the previous experiment in Section 4.2 but with projection dimensions $m = \gamma d$ for $\gamma \in \{1, 2, 4, 8, 16\}$. For every choice of m we then constructed an SJLT with $s \in \{1, 2, 4, 8, 16\}$. We measured both the summary time (i.e to construct SA) in seconds and the empirical error shown as the mean Frobenius norm over 10 trials. Figure 2 illustrates the behavior of this error. Unsurprisingly, and consistent with the theory, as γ is increased, we see that the sketching error is lower. However, it is striking that a relatively substantial increase in the column sparsity does not affect the error too much: the error behaviour looks almost constant with only mild changes as s varies in either direction.

4.4 Summary Construction Speedup

To better understand the overall time cost, we measure the time taken to build the necessary summaries for each γ. Figure 3 plots the "summary speedup" when using the CountSketch rather than the SRHT, averaged over ten repetitions for each γ value. This shows that our methods offer a greater speedup when the data is both large and sparse. Particular standouts are the Abtaha data (150x faster on average), and the Specular and w8a datasets (around a 100x average speedup compared to the SRHT).

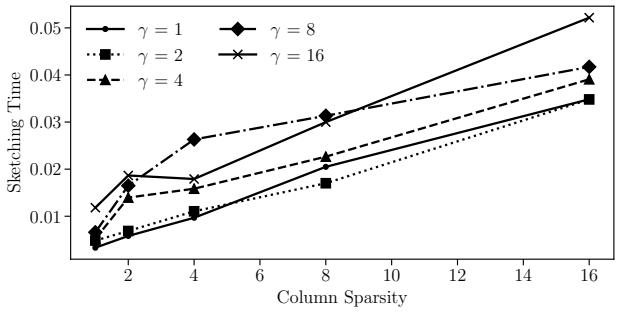


Figure 4: Sketch time vs column sparsity s on w4a

The *fraction of succesful embeddings* was also measured: this is the number of times out of 10 trials for which a full rank embedding was returned. Under our assumption on the size of n and d, it would not be feasible to compute an SVD in order to check rank(A) in practice. However, for the purpose of this study we compute the rank offline and use it to check if rank(SA) = rank(A). We performed 120 trials (10 trials on 12 datasets) and found that the Gaussian sketch was the best performing, as it retained rank 100% of the time. The SJLT retained rank in at least 99% of the trials.[5] The SRHT and CountSketch each returned correct embeddings at least 95% of the time. Interestingly, for $\gamma > 1$ and $s > \gamma$, we still retain rank for the vast marjority of the time using the SJLT. However, losing rank is not fatal under the IHS framework: it just means that in one iteration, we fail to find the best direction to make progress in. However, since iterations are independent, this can be made up in a subsequent step. Overall, our calibration experiments and IHS experiments vindicate these choices of projection dimensions and column sparsity on real datasets.

As expected, Figure 4 shows a broadly linear increase in the time to sketch the data as s is increased: this is consistent with the theory as SJLT uses $O(s \cdot \text{nnz}(A))$ time to apply the transform. The time taken to apply the sketch seems to increase more rapidly with the projection dimension rather than the column sparsity. This is indicated by the general trend that increasing γ shifts the time-sparsity curve upwards by a small constant. For very sparse sketches ($s = 1$), the time to sketch this data is consistently below 0.01s and there is a small absolute difference yet for a fixed s. The sketch times increases monotonically in m over all datasets.

5 ORDINARY LEAST SQUARES RESULTS

In this section, we implement the sparse embeddings in the IHS framework for the first time, and show that it offers many advantages over alternative sketching methods. When IHS was introduced in [20], it was evaluated primarily using (dense) Gaussian sketches, with some experiments separately adopting SRHT. Our aim in this evaluation is to compare the impact of the choice of sketch method within the framework on the time cost, across a range of different data sizes and densities. In order for a genuine cross-comparison of all four sketch types (CountSketch, SJLT, SRHT and Gaussian), we replicate the set up of Pilanci and Wainwright [20] for OLS regression. We then repeat the experiment by drawing SJLT sketches with various values of s to see if column sparsity has a substantial impact. The experiments in this section use synthetic data $A \in \mathbb{R}^{n \times d}$ whose entries are drawn from a standard normal distribution. A ground truth vector x^* is chosen at random and then standard Gaussian noise is added ω_i for $i = 1, \ldots, n$. The target vector is then $b = Ax^* + \omega$.

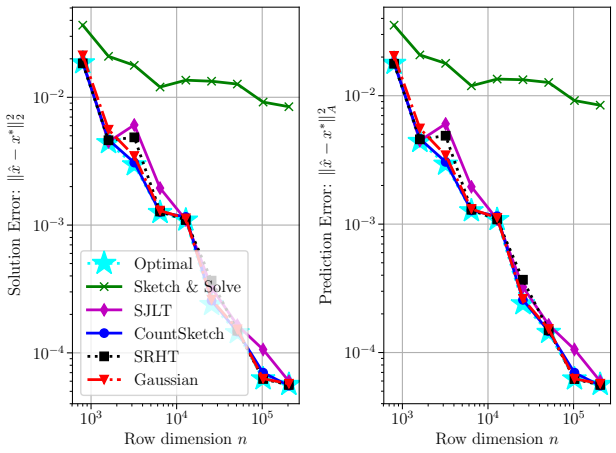
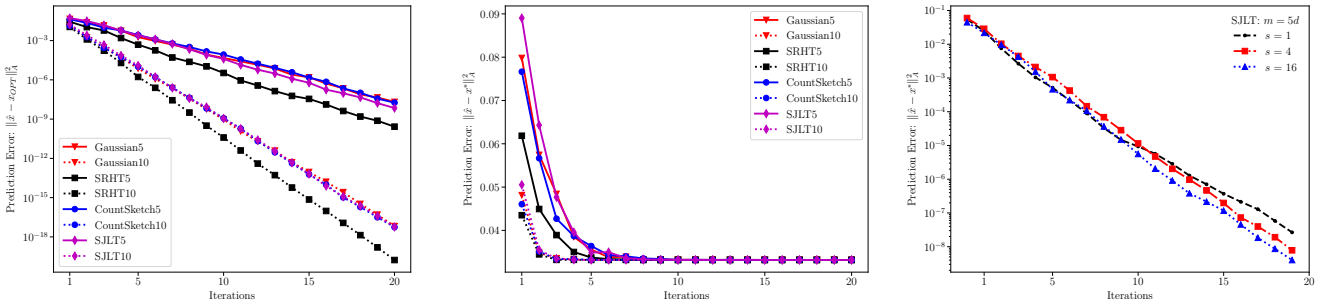


Figure 5: Solution and prediction errors as n varies

Error compared to data size. (Figure 5) The first experiment seeks to understand how the error responds as n increases. For this task we examine how well our estimated weights $\hat{x}$ match the error performance of x_{opt} when compared to the *ground truth* x^*. Data $A \in \mathbb{R}^{n \times d}$ is generated for $n \in \{100 \times 2^i \text{ for } i \in \{3, 4, \ldots, 12\}\}$ The number of iterations is fixed at $N = 1 + \log n$ and the sketch size for IHS is fixed at $m = 5d$. Each experiment is repeated 10 times and the mean error is reported. We show the results for IHS with the four different sketch types, and for an idealized "optimal" version which computes the exact solution x_{opt} to Equation (1). Note that solving Equation (3) with $S = I$, i.e. performing one step of the iterations with no sketching is equivalent to finding x_{opt} exactly, so we implement this as a direct solve on the original instance. We also instantiate the S&S model of approximation with sketch dimension set to $m' = Nm$,

[5] We ignore the setups when $\gamma = 1$ and $s > 1$ for the SJLT, which corresponds to the concatenation of CountSketches with fewer than d rows, wherein rank cannot possibly be retained.



(a) Error to optimal vs iterations **(b) Error to truth vs iterations** **(c) Error vs # Iterations: SJLT $m = 5d$.**

Figure 6: Solution & ground truth error for different sketch sizes

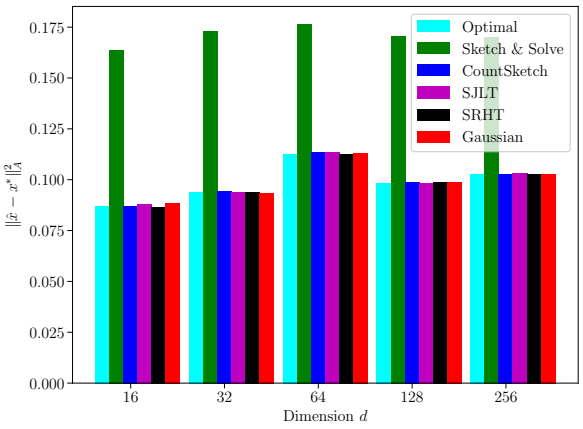


Figure 7: Prediction errors as d increases

(i.e. equal to the sum of sketch sizes in IHS) and the CountSketch as the projection method. However, consistent with the theory, sketch and solve is observed to be significantly weaker than other methods. The IHS methods all give very similar behavior: as the number of samples n increases, the errors to the statistical problem of estimating x^* decrease in line with the theory (reciprocal in n, per Proposition 3.2), and secondly, the iterative sketching essentially matches the optimal (non-sketching) estimator. The number of iterations increases very slowly with n, so the cause of this can be attributed to the increased evidence available from the input. There is no notable difference in the accuracy performance of the four sketch types, although SJLT appears to be slightly more variable. However, this difference from the variation is very small and each sketch method used in IHS is favorable to the sketch-and-solve estimator from this perspective.

Solution Error vs Number of Iterations. (Figures 6a and 6b) We next study the convergence properties of IHS. We sampled OLS instances as above with $(n, d) = (6000, 200)$ and tested sketch sizes $m = \gamma d$ for $\gamma = 5, 10$. The IHS method was run for $T = 1, 2, \ldots, 20$ iterations and for each different sketch the error to the optimal estimator was measured (averaged over 10 repeats). Figures 6a and 6b show the error against the instance optimal solution and ground truth on a logarithmic and linear scale, respectively. Each group of lines (dotted and solid) shows results for different sketch sizes. All methods converge geometrically towards the optimal solution with the rate increasing when larger γ is used (Figure 6a). Larger sketch sizes clearly help convergence speed as a function of iterations. We see that SRHT has slightly faster convergence behavior as a result of larger per-step error reduction. However, while the error to the optimal estimator quickly becomes vanishingly small, there is a larger cut-off in accuracy to the ground truth, as shown in Figure 6b. This cut-off is determined solely by the dimensions of the input, which in this example is computed as $200/6000 \approx 0.03$ Here, we see that larger sketches help to reach this point, and that SRHT is mildly faster for larger sketches but is noticeably faster by one or two iterations on average for smaller sketches. This difference between the SRHT and other sketches is further exaggerated for smaller m (omitted for legibility).

While Figures 6a and 6b are useful in cross-sketch comparison, we need to understand how sparse embeddings with different column sparsities behave in the iterative framework. Following the calibrations discussed in Section 4.3, we fix $m = 5d$ and test sparse embeddings with column sparsity $s = 1, 4, 16$. For $s = 1$ we have the vanilla CountSketch which is fast but less accurate and has a small chance (but noticeably higher than for $s > 1$) of returning an embedding with high error compared to other methods. At the other extreme, setting $s = 16$ is predicted to be much more accurate but take longer to apply. In between, we choose $s = 4$, and show the results for $m = 5d$ in Figure 6c, averaged over 10 repetitions. We see very little variation in accuracy over the different choices of s, in line with the results in Figure 2. There is more

variability from CountSketch ($s = 1$), which tends to finish on a slightly worse estimate. However, overall CountSketch outperforms its worst case guarantees, and is competitive with more theoretically robust sparse embedding sketches.

Error compared to dimensionality. (Figure 7) We generate data with $d \in \{16, 32, 64, 128, 256\}$ and take $n = 250d$ so that the aspect ratio is fixed at $1/250$. We use $T = 1 + \log n$ iterations in the IHS and $m = 10d$; hence the corresponding S&S sketch dimension is $T\gamma d$. The results shown in Figure 7 are averaged over 10 repetitions. IHS is instantiated using each of the CountSketch, SJLT, SRHT and Gaussian sketches while S&S uses CountSketch. When n is smallest, i.e for $d = 16, 32$, the sparse methods appear to perform a (very small) constant factor worse than the SRHT but remain competitive to the Gaussian methods. However, this difference becomes negligible when n increases. For every $d \geq 64$, IHS with CountSketch or SJLT performs comparably to both the more expensive methods. In all instances, the S&S estimate is noticeably worse at estimating the solution error than any of the IHS methods, consistent with Figure 5.

6 LASSO RESULTS

In what follows, we write RP(γ) to denote the random projection method RP with projection dimension $m = \gamma d$. For example, SRHT(10) is an SRHT projection with projection dimension $m = 10d$.

Experimental Setup. We fix $\lambda = 5.0$ and use this to define an instance of LASSO regression $x_{OPT} = \text{argmin}_{x \in \mathbb{R}^d} \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1$. This choice of λ ensures that all solutions x_{OPT} are bounded away from zero so that the algorithm cannot report zero as a viable approximation. Each iteration builds a smaller quadratic program (in terms of n) which is solved exactly, akin to [13]. Due to the speedups observed in Figure 1b, we measure error against wall-clock time. We tested 10 independent runs of the algorithm and plot mean error. To enable the fairest comparison between the sparse sketches and SRHT, we uniformly subsample $n' = 2^{\lfloor \log_2(n) \rfloor}$ rows prior to beginning each trial[6].

Our choice of projection dimension m is guided by Theorem 3.12 (stating that we require $\varepsilon < \frac{1}{2}$) and the calibration experiments from Section 4.2 which suggest appropriate accuracy can be obtained when $m \geq 5d$. The SJLT was initialised with column sparsities $s = 1$ and $s = 4$ and we compare to SRHT and Gaussian transforms. Experiments for $m < 5d$ (e.g. $m = 4d$ in Figure 6a) were slow to converge, or diverged, similar to results in [31]. Thus, we used $m = 5d, 10d$.

[6] Note that only the SRHT requires n to be a power of 2, so the differences between sparse embeddings and the SRHT are more pessimistic here than for general n.

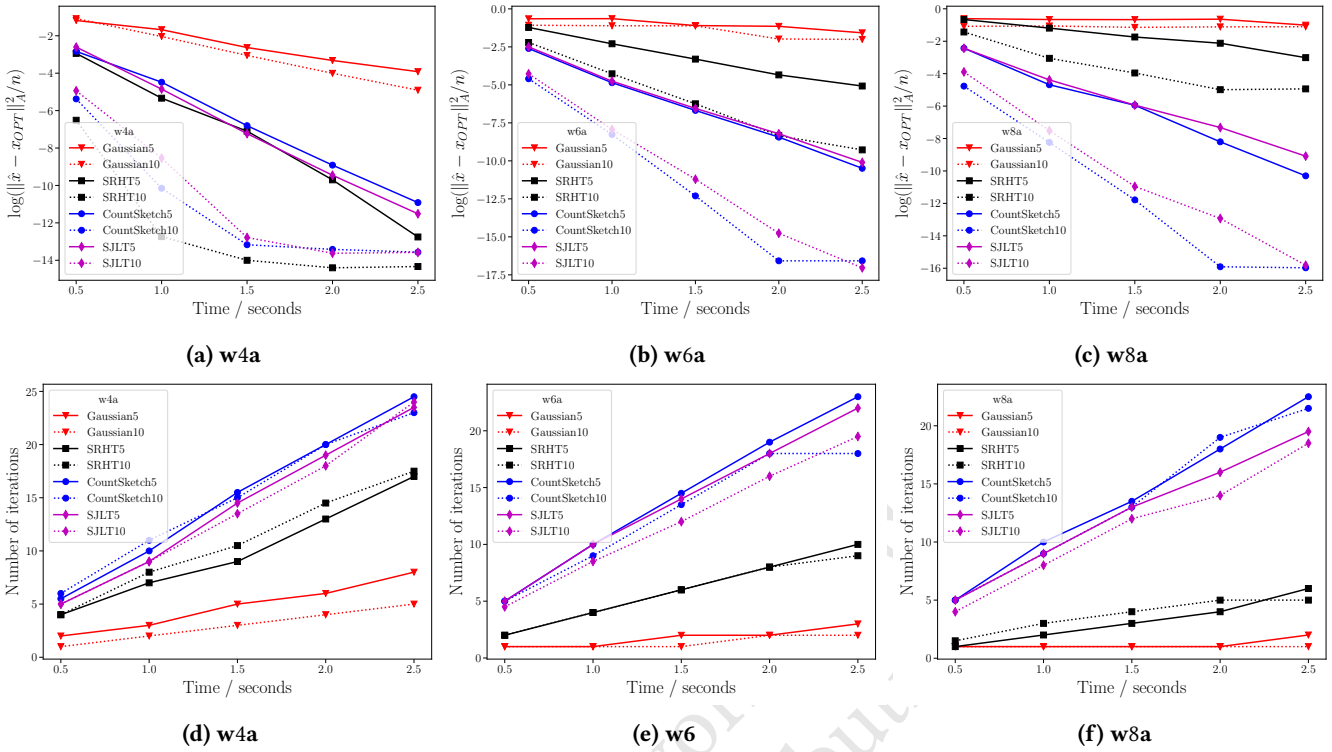
Real Datasets. Results are given for a selection of 6 real datasets in Figures 8 and 9. These both plot the error achieved over duration of the experiment (top row of each plot), and the corresponding progress in terms of number of iterations in the same time (bottom row). Additional experiments on synthetic data led to similar conclusions, and are omitted for space reasons.

The first observation is that faster convergence is seen when sketch sizes are larger. Recall that sparse sketches have only a mild increase in the summary time when the projection dimension m is increased as seen in Figures 1b and 4. Hence, one should aim to tradeoff with making the sketch as large as possible while not inhibiting the number of iterations that can be completed. For the smallest sketch size $m = 5d$, there is less difference between the competing methods. For larger sketches, CountSketch is much faster to reduce error. In Figures 8 and 9 we see that using CountSketch allows more iterations in the same time. While sketch size has limited impact on iteration cost, larger sketches have lower error so descend to the optimum faster.

Error for increasing n. The wna ($n = 4, 6, 8$) datasets are taken from a single dataset and are training sets of increasing size [22]. The findings for the smallest sparse dataset, w4a, are shown in Figure 8a. Although this dataset is sparse, the SRHT appears favourable. The w4a dataset is small enough that converting it to a dense array is still memory-efficient and the dense random projection can be applied. Figure 8d shows that the SRHT completes slightly fewer iterations than both sparse methods; the absolute difference on average being between 5 and 10 fewer at any given time.

We see behavior consistent with Figure 6a in that more progress is made in fewer iterations using the SRHT (when this is possible). Additionally, we begin to see the deficiencies of the Gaussian random projection already on an instance with only 4096 rows: the cost of dense matrix multiplication is prohibitive in comparison to the SRHT. In contrast, both of the sparse methods with $m = 10d$ achieve similar error to SRHT(10), yet at $m = 5d$ are not quite as accurate as SRHT(5). However, both sparse methods are comfortably superior to the Gaussian projection at $m = 5d, 10d$.

While on the w4a dataset it is difficult to see the benefits of the sparse random projections, when the size of the dataset is increased, these benefits become apparent. On w6a (Figure 8b) we begin to see a marked improvement in the error behaviour of the sparse sketches at a projection dimension of $m = 10d$. At $m = 5d$ the sparse sketches have comparable error performance as the SRHT(10), albeit with at least twice as many iterations completed as seen in Figure 8e. Again we see the Gaussian projection flounder, completing at most 3 iterations and making essentially no progress towards the



(a) w4a (b) w6a (c) w8a

(d) w4a (e) w6 (f) w8a

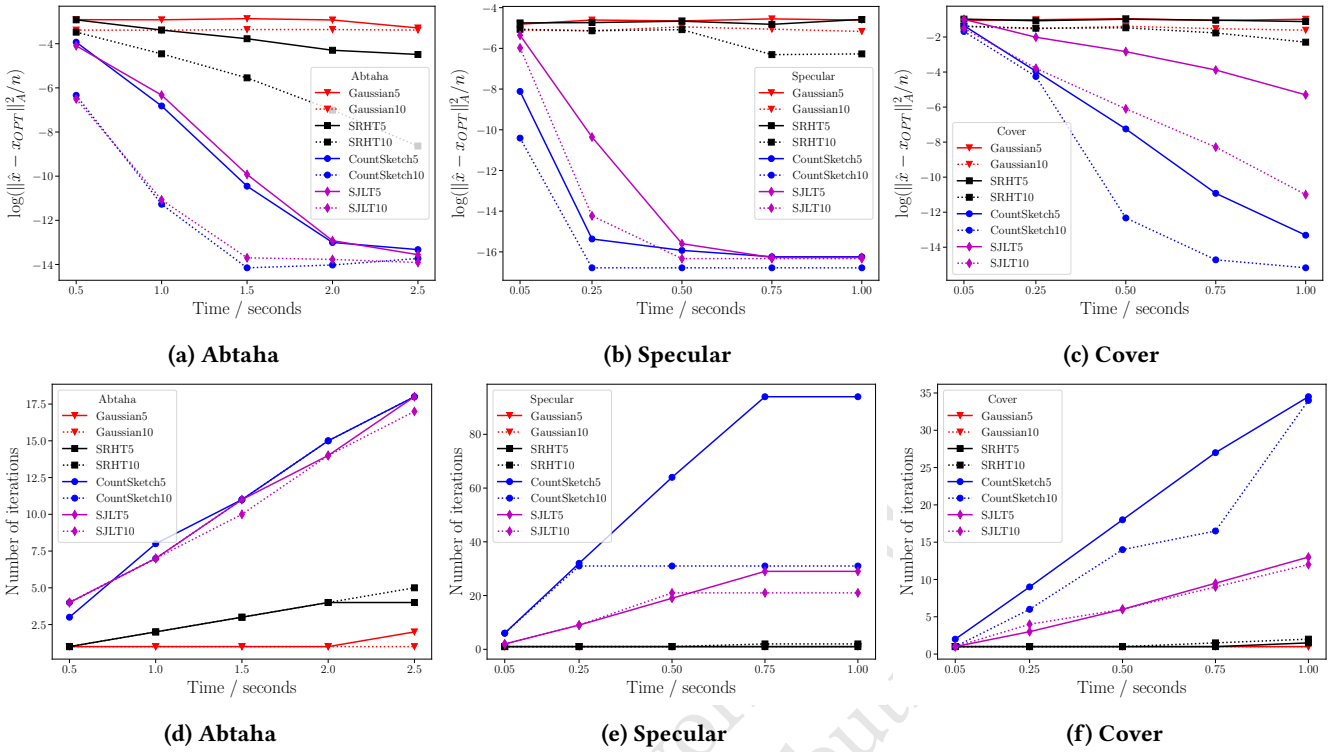
Figure 8: IHS Methods: Solution Error vs Time on *wna*

solution before the competing methods have completed. Although it still seems that the SRHT is somewhat competitive, increasing n once more to obtain the w8a dataset shows that using the SRHT becomes uncompetitive on large sparse datasets. Indeed, in Figure 8c we see that the sparse sketches with $m = 10d$ achieve error below 10^{-10} in 1.5 seconds. This is now a clear and substantial improved error performance in comparison to the SRHT whose error at either $m = 5d$ or $m = 10d$ has not progressed much (if at all) beyond 10^{-5}. This improvement can be attributed to the increased number of iterations completed using the sparse sketches: in 1.0 seconds, roughly 1 or 2 iterations were completed (on average) using SRHT(5) or SRHT(10). In comparison, both sparse methods had completed over 10 iterations in the same time frame for both projection dimensions so much more progress towards the optimum had been achieved.

We compare the results on the w8a dataset to those on the Abtaha dataset (in Figure 9) which was chosen due to its comparable size and sparsity as w8a. There is a slight reduction in the number of iterations because Abtaha has 30 more columns than w8a and solving the intermediate QPs scale with $\Omega(md^2)$ just to compute $(SA)^T(SA)$. This scaling is not too severe, however, and the efficacy of both sparse embeddings is clear with the picture and conclusions remaining much the same.

Large and sparse datasets. To see how the sketches perform as we encounter very large and sparse data we also repeat the experiment on the Specular dataset. We constrained time even more and tested at intervals $[0.05, 0.25, 0.5, 0.75, 1.0]$ seconds but kept the same penalty parameter $\lambda = 5.0$.

Figures 9b and 9e show that yet again the sparse embeddings dominate the dense projections in terms of error. Rapid convergence is seen with CountSketch(10) which reaches machine precision and terminates in 0.25 seconds. Very similar behaviour is seen with CountSketch(5) as almost all of the progress is made in 0.25 seconds. The SJLT(10) performs similarly to CountSketch(5) in terms of error decay. However, when $m = 5d$ we begin to notice a slight deterioration in performance of the SJLT(5) compared to CountSketch(5) and CountSketch(10): the marginally increased time costs now begin to play a role as the CountSketches have completed more iterations and hence made further progress. Similarly, the SJLT(5) incurs slightly higher per-iteration error than the SJLT(10) so less progress is made in the allotted time. In spite of this, both CountSketch(·), SJLT(·) perform *significantly* better than the dense methods for which the data is so large that operating on the dense arrays becomes infeasible. Only the SRHT(10) begins to make any substantial progress, completing 2 iterations in 0.75 seconds on average which obtains error roughly 10^{-6}. Accordingly, reasonable progress



(a) Abtaha (b) Specular (c) Cover

(d) Abtaha (e) Specular (f) Cover

Figure 9: IHS Methods: Solution Error vs Time (sparse data).

has been made toward the optimum due to the high-quality per-iteration performance of the SRHT (again, cf. Fig 6a) yet the price to pay for this is a drastically increased time cost.

To check our assertions we also tested on the CoverType dataset. We selected this dataset as it is of comparable size to the Specular dataset (in both n and d) while remaining less than 50% dense. Results for this experiment are given in Figures 9c and 9f. The general trend remains the same, however, we notice a reduction in the number of iterations completed in the allotted time across all methods but most clearly for the sparse embeddings compared to Figure 9e. This is attributable to the increased number of nonzeros in CoverType compared to Specular. In spite of this, the performance of sparse embeddings remains superior to the dense counterparts which again fail to make any progress.

7 CONCLUSION

We have studied theoretical and practical considerations for performing regression on large scale data via matrix embeddings. We have shown the Iterative Hessian Sketch (IHS) framework can be used effectively for scenarios with large n, and $n \gg d$, using fast sparse embeddings. A consequence of this is much faster descent towards the optimal solution than the state of the art, on data both sparse and dense. CountSketch tends to be the overall fastest method to converge on a solution, despite having weaker theoretical guarantees. SJLT has to do a bit more work per iteration to build the sketch, and so the saving in the number of iterations to converge is not quite enough to compensate.

There are several directions for future work. First, it would be useful to generalize from cases where the regression objective is a more general class of convex loss functions, rather than least-squares (as in (1)). This has been initiated for dense sketches, but is not known for sparse embeddings [21]. A second direction is to reduce the need for access to the data. The current algorithm builds a new sketch of the data in each iteration, based on the current solution. It would be desirable to relax this, to enable deployment in (streaming) systems which only have one pass over the data. S&S already meets this requirement, but does not obtain a high enough level of accuracy. Recent theoretical work has shown this possible for the case of ridge regression (least squares with a quadratic regularization term) [6] but full access to the data is required for the inner product term of Equation (3).

A SPECTRAL PROPERTIES OF COUNTSKETCH

We must show that for a constant η, independent of m and n:

$$\|\mathbb{E}\left[S^T(SS^T)^{-1}S\right]\|_2 \leq \eta\frac{m}{n}. \tag{10}$$

We start with a lemma on the behavior of Bernoulli random variables which supports the analysis.

Lemma A.1. Let $X = \sum_{i=1}^{n} X_i$ where each X_i is an iid Bernoulli random variable with parameter p. Then

$$\mathbb{E}\left[\frac{1}{1+X}\right] = \frac{1-(1-p)^{n+1}}{(n+1)p} \tag{11}$$

Proof. Observe that X is a sum of Bernoulli rvs and so shares the Binomial probability mass function, hence:

$$\mathbb{E}\left[\frac{1}{1+X}\right] = \sum_{k=0}^{n}\frac{1}{1+k}\binom{n}{k}p^k(1-p)^{n-k} \tag{12}$$

$$= \frac{1}{p(n+1)}\sum_{k=0}^{n}\binom{n+1}{k+1}p^{k+1}(1-p)^{n-k} \tag{13}$$

$$= \frac{1-(1-p)^{n+1}}{p(n+1)}. \tag{14}$$

where the last equality holds by the Binomial Theorem: we have a (scaled) sum over the full Binomial distribution excepting the first term, $\binom{n+1}{0}p^0(1-p)^{n+1} = (1-p)^{n+1}$. □

Theorem A.2. Let S be an $m \times n$ CountSketch matrix with no all-zero rows. The spectral bound (10) is met with $\eta = 1$.

Proof. We analyse the quantity $\mathbb{E}\left[S^T(SS^T)^{-1}S\right]$ and show this is a diagonal matrix whose entries are a constant multiple of m/n. First we deal with the term SS^T which by Lemma 3.6 is a diagonal matrix whose entries are N_i, the number of nonzeros in row S_i. Hence we write $D = SS^T$ with entries $D_{ii} = N_i$ and off-diagonal entries zero. Since by our assumption all the diagonal entries in D are non-zero, D^{-1} exists and is has $D^{-1}_{ii} = 1/N_i$. Now, we write $S^T(SS^T)^{-1}S = S^TD^{-1}S$ and distribute entries of D^{-1} as $S^TD^{-1/2}D^{-1/2}S$, which is well-defined as D^{-1} is both diagonal and positive. Set $\bar{S} = D^{-1/2}S$, which is simply S but scaled by its row norms as:

$$\bar{S}_{ij} = \frac{S_{ij}}{N_i^{1/2}}. \tag{15}$$

This expresses our object of interest as the matrix $\bar{S}^T\bar{S}$, which is the collection of scaled inner products between the columns of S. In particular, for every column $\bar{S}^j$ the unique nonzero entry is located at $\bar{S}_{h(j),j}$ and takes value $S_{h(j),j}/N_{h(j)}^{1/2}$.

Since $\bar{S}$ has the same sparsity structure as S and its entries are just rescaled versions of those in S, we know from Lemma 3.8 that in expectation the only entries we need consider are those on the diagonal. Then the inner product to consider is $\langle\bar{S}^i, \bar{S}^i\rangle$ which is exactly the norm of column $\bar{S}^i$. Equation (15) means that $(\bar{S}^T\bar{S})_{ii} = 1/N_{h(i)}$ and Lemma A.1 can be used to bound $1/N_{h(i)}$ in expectation as it is a sum of Bernoulli random variables. Indeed, $N_{h(i)} = \sum_{j=1}^{n}\mathbf{1}(S^2_{h(i),j})$ where $\mathbf{1}(S_{h(i),i})$ is an indicator variable for the presence of a nonzero in entry $S_{h(i),j}$. No row of S is identically zero so certainly $S^2_{h(i),i} = 1$ and for any $j \neq i$ then $\mathbf{1}(S_{h(i),j}) = 1$ with probability $1/m$; for a given column, non-zero rows are chosen uniformly at random in the CountSketch construction. This is exactly when $h(j) = h(i)$ so:

$$N_{h(i)} = \sum_{j=1}^{n} S^2_{h(i),j} = 1 + \sum_{\substack{j\neq i \\ h(j)=h(i)}} S^2_{h(i),j}. \tag{16}$$

Hence, we may write $N_{h(i)} = 1 + X$ and invoke Lemma A.1 with $p = 1/m$ and $n-1$ trials (since there are $n-1$ columns columns $j \neq i$) to obtain:

$$\mathbb{E}\left[\frac{1}{1+X}\right] = \frac{1-(1-\frac{1}{m})^n}{n/m} \leq \frac{1}{n/m} = \frac{m}{n}$$

Thus, $\mathbb{E}\left[\bar{S}^T\bar{S}\right]_{ij} \leq m/n$ if $i = j$ and is zero otherwise. That is, $\mathbb{E}[S^T(SS^T)^{-1}S] \leq \frac{m}{n}I_n$ so (10) holds with $\eta = 1$, and the spectral norm is bounded above by m/n □

REFERENCES

[1] [n. d.]. Apache Kafka a distributed streaming platform. https://kafka.apache.org/.

[2] [n. d.]. Apache Mahout: Scalable machine learning and data mining. http://mahout.apache.org/.

[3] [n. d.]. Machine Learning Library (MLlib). http://spark.apache.org/mllib/.

[4] Nir Ailon and Bernard Chazelle. 2006. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. ACM, 557–563.

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

[6] Agniva Chowdhury, Jiasen Yang, and Petros Drineas. 2018. An iterative, sketching-based framework for ridge regression. In *International Conference on Machine Learning*. 988–997.

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

[8] Kenneth L Clarkson and David P Woodruff. 2013. Low Rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 81–90.

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

[10] Yogesh Dahiya, Dimitris Konomis, and David P Woodruff. 2018. An Empirical Evaluation of Sketching for Numerical Linear Algebra. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1292–1300.

[11] Timothy A Davis and Yifan Hu. 2011. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)* 38, 1 (2011), 1.

[12] Dua Dheeru and Efi Karra Taniskidou. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[13] Brian R Gaines, Juhyun Kim, and Hua Zhou. 2018. Algorithms for fitting the constrained lasso. *Journal of Computational and Graphical Statistics* just-accepted (2018).

[14] Jiawei Jiang, Fangcheng Fu, Tong Yang, and Bin Cui. 2018. SketchML: Accelerating distributed machine learning with data sketches. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 1269–1284.

[15] Daniel M Kane and Jelani Nelson. 2014. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)* 61, 1 (2014), 4.

[16] Daniel M Kane, Jelani Nelson, and David P Woodruff. 2010. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1161–1178.

[17] Daniel M Kane, Jelani Nelson, and David P Woodruff. 2010. An optimal algorithm for the distinct elements problem. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 41–52.

[18] Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, Saravanan Thirumuruganathan, Sanjay Chawla, and Divy Agrawal. 2017. A cost-based optimizer for gradient descent optimization. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 977–992.

[19] Andrew McGregor, A Pavan, Srikanta Tirthapura, and David Woodruff. 2012. Space-efficient estimation of statistics over sub-sampled streams. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. ACM, 273–282.

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

[21] Mert Pilanci and Martin J Wainwright. 2017. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization* 27, 1 (2017), 205–245.

[22] John Platt. 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. (1998).

[23] John C Platt. 1999. 12 fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods* (1999), 185–208.

[24] Eric Price, Zhao Song, and David P Woodruff. 2017. Fast Regression with an ℓ_∞ Guarantee. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP 2017)*.

[25] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*. 693–701.

[26] Tamas Sarlós. 2006. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, 143–152.

[27] Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. 2016. Learning linear regression models over factorized joins. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 3–18.

[28] Kai Sheng Tai, Vatsal Sharan, Peter Bailis, and Gregory Valiant. 2018. Sketching linear classifiers over data streams. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 757–772.

[29] Lieven Vandenberghe. 2010. The CVXOPT linear and quadratic cone program solvers. *Online: http://cvxopt. org/documentation/coneprog. pdf* (2010).

[30] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations* 15, 2 (2013), 49–60. https://doi.org/10.1145/2641190.2641198

[31] Jialei Wang, Jason D Lee, Mehrdad Mahdavi, Mladen Kolar, Nathan Srebro, et al. 2017. Sketching meets random projection in the dual: A provable recovery algorithm for big and high-dimensional data. *Electronic Journal of Statistics* 11, 2 (2017), 4896–4944.

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

[33] David P Woodruff and Qin Zhang. 2018. Distributed Statistical Estimation of Matrix Products with Applications. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. ACM, 383–394.