Iterative Hessian Sketch in Input Sparsity Time

Graham Cormode & Charlie Dickens
University of Warwick

Summary

Large-scale ML systems require scalable algorithms to approximately solve convex constrained least squares (CCLS) problem.

Iterative Hessian Sketch (IHS) is a randomized approximate Newton-type method to approximate solutions of these problems.

We prove that *Sparse Johnson Lindenstrauss Transforms* (SJLT) can be used in IHS and inherit the performance guarantees.

Our empirical evaluation shows that in practice the SJLT methods perform very well.

Surprisingly, we find that the weaker CountSketch performs better than the theory predicts in this setup.

Our approach can summarize data roughly *100x faster* for sparse data, and, surprisingly, *10x faster* on dense data.

The SJLTs are most beneficial when data is large $(n \gg d)$ and sparse.

Problem Setup

Problem: Input: data $A \in \mathbb{R}^{n\times d}$, target vector $b \in \mathbb{R}^n$ and convex constraints $\mathcal{C} \subseteq \mathbb{R}^d$.

We assume that $b = Ax^* + \omega$ with $\omega_i \sim N(0, \sigma^2)$, $n \gg d$, and would like to obtain the solution to the CCLS:

$$x_{\text{opt}} = \text{argmin}_{x\in\mathcal{C}}\, f(x), \qquad f(x) = \tfrac{1}{2}\|Ax - b\|_2^2. \tag{1}$$

Obtaining x_{opt} in (1) takes $O(nd^2)$ time to generate the Hessian matrix $\nabla^2 f(x) = A^T A$.

Instead of solving (1) exactly, we approximate x_{opt} prediction (semi)norm $\|x\|_A = \frac{1}{\sqrt{n}}\|Ax\|_2$ through a sequence of Newton-type iterates.

Iterative Hessian Sketch uses random projections to approximate $A^T A$ for iterates.

Estimates x^{t+1} gradually descend towards x_{opt} via steps defined by (2) using random projection $S^j \in \mathbb{R}^{m\times n}$ with $m \ll n$.

$$x^{t+1} = \underset{x\in\mathcal{C}}{\text{argmin}}\, \tfrac{1}{2}\|S^{t+1}A(x - x^t)\|_2^2 - \langle A^T(b - Ax^t), x - x^t\rangle. \tag{2}$$

Theorem. [1] Given $A, b, \mathcal{C}$ which define a CCLS problem, the IHS approach for t iterations returns an estimate $\hat{x}$ which approximates x_{opt} according to: $\|\hat{x} - x_{\text{opt}}\|_A \leq \varepsilon^t \|x_{\text{opt}}\|_A$.

We show that iterations can be completed in **input sparsity time**.

Background and Baselines

Definition. 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 $x \in \mathbb{R}^d$, $\|SAx\|_2^2 \in [(1-\varepsilon)\|Ax\|_2^2, (1+\varepsilon)\|Ax\|_2^2]$.

Various random projections including the Gaussian and Subsampled Hadamard (SRHT) transforms can define subspace embeddings. Our proposal is to adopt **sparse embeddings**:

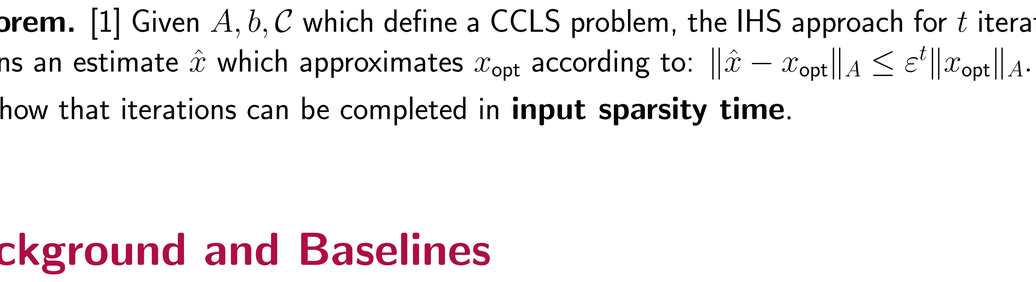
CountSketch: Initialise $S = \mathbf{0}_{m,n}$ and for every column i of S choose a row $h(i)$ uniformly at random. Set $S_{h(i),i}$ to either $+1$ or -1 with equal probability.

Sparse Johnson-Lindenstrauss Transform (SJLT): 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 $m/s \times n$.

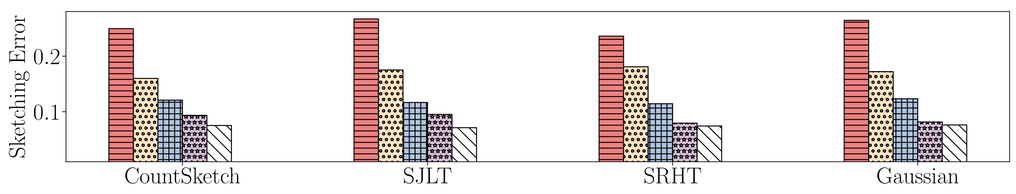
Sketch Method	Embedding Dimension (m)	Projection Time
Gaussian	$O\left(d\varepsilon^{-2}\log\frac{1}{\delta}\right)$	$O(nd^2)$
SRHT	$O\left(d\varepsilon^{-2}\log\frac{1}{\delta}\right)$	$O(nd\log n)$
CountSketch	$\boldsymbol{O\left(d^2/\delta\varepsilon^2\right)}$	$\boldsymbol{O}(\text{nnz}(\boldsymbol{A}))$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

Tab. 1: Time/Space Complexity for Subspace Embedding

Baselines. We measure the time to sketch $n \mapsto m$ for projection dimension $m = \gamma d$.



Sketch time for CountSketch at the same projection dimension is favourable, but is the cost of this an increased error?



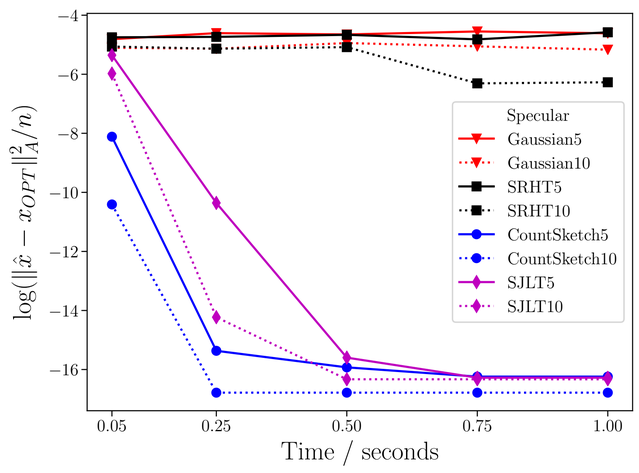
When $n \gg d$ the error performance of CountSketch is better than the theory predicts at $m = \gamma d$.

Experimental Evaluation

Experimental Setup. We fix $\lambda = 5.0$ and use this to define an instance of LASSO regression $x_{\text{opt}} = \text{argmin}_{x\in\mathbb{R}^d} \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1$.

The SJLT was initialised with column sparsities $s = 1$ (CountSketch) and $s = 4$. We compare to SRHT and Gaussian transforms. Sketches with projection dimension $m = \gamma d$ are denoted sketch(γ)

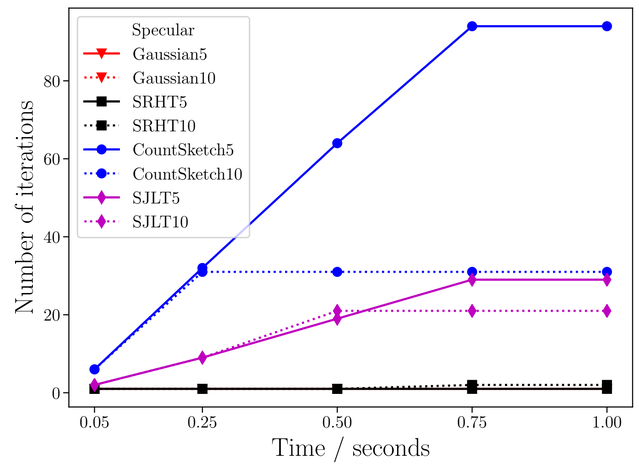
Experimental Findings. On large sparse data, SJLTs outperform the error reduction of the dense counterparts



Error for sparse methods decays significantly more quickly in wall-clock time to x_{opt} than Gaussian or SRHT

The SJLT$(\cdot)$ performs similarly to CountSketch(5) in terms of error decay but the marginally increased *sketch time* results in slightly less overall progress.

CountSketch(5) is a weaker embedding than CountSketch(10) so needs significantly more iterations in a given time to reach comparable error.



In summary, many cheap coarse embeddings enable convergence much quicker than expensive embeddings.

Conclusions and Future Work

Despite being a weaker sketch in theory, CountSketch tends to give fastest convergence.

SJLT needs slightly more work per iteration to build the sketch yet the saving in the number of iterations to converge is not quite enough to compensate versus the weaker CountSketch.

How do we reduce the need for access to the data? Ideally want a single pass over the data, perhaps with weaker error guarantee.

Code & Full paper: `https://github.com/c-dickens/sketching_optimisation`

Includes technical details, lower bound for sketch-and-solve model suboptimality, speedup evaluation, comparison of sparsity parameters for SJLT, validation on more datasets.

Workshop version: `https://arxiv.org/abs/1910.14166`

References

[1] Mert Pilanci and Martin J. Wainwright. "Iterative Hessian sketch: Fast and Accurate solution approximation for constrained least-squares". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1842–1879.

[2] David P. Woodruff. "Sketching as a tool for numerical linear algebra". In: *Foundations and Trends® in Theoretical Computer Science* 10.1–2 (2014), pp. 1–157.

Acknowledgments

GC and CD acknowledge support from the European Research Council grant ERC-2014-CoG 647557 & The Alan Turing Institute under EPSRC grant EP/N510129/1.