John Ellipsoids via Lazy Updates

David P. Woodruff
Carnegie Mellon University
dwoodruf@cs.cmu.edu

Taisuke Yasuda
Voleon Group
yasuda.taisuke1@gmail.com

Abstract

We give a faster algorithm for computing an approximate John ellipsoid around n points in d dimensions. The best known prior algorithms are based on repeatedly computing the leverage scores of the points and reweighting them by these scores [CCLY19]. We show that this algorithm can be substantially sped up by delaying the computation of high accuracy leverage scores by using sampling, and then later computing multiple batches of high accuracy leverage scores via fast rectangular matrix multiplication. We also give low-space streaming algorithms for John ellipsoids using similar ideas.

1 Introduction

The *John ellipsoid* problem is a classic algorithmic problem, which takes as input a set of n points $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\}$ in d dimensions, and asks for the minimum volume enclosing ellipsoid (MVEE) of these n points. If P is the convex hull of these n points, then a famous result of Fritz John [Joh48] states that such an ellipsoid satisfies $\frac{1}{d}Q \subseteq P \subseteq Q$, and if P is furthermore symmetric, then $\frac{1}{\sqrt{d}}Q \subseteq P \subseteq Q$. Equivalently, we may consider the n input points to be constraints of a polytope $\{\mathbf{x} : \langle \mathbf{a}_i, \mathbf{x} \rangle \leq 1, i \in [n]\}$, in which case the problem is to compute a maximal volume inscribed ellipsoid (MVIE). These two problems are related by taking polars, which corresponds to inverting the quadratic form defining the ellipsoid. In this work, we focus on the symmetric case, so that the polytope P may be written as $P = \{\mathbf{x} : \|\mathbf{A}\mathbf{x}\|_\infty \leq 1\}$, where $\mathbf{A}$ denotes the $n \times d$ matrix with the n input points $\mathbf{a}_i$ in the rows, and our goal is to output an ellipsoid Q which approximately satisfies $Q \subseteq P \subseteq \sqrt{d} \cdot Q$.

The John ellipsoid problem has far-reaching applications in numerous fields of computer science. In statistics, the John ellipsoid problem is equivalent to the dual of the D-optimal experiment design problem [Atw69, Sil13], in which one seeks weights for selecting a subset of a dataset to observe in an experiment. Other statistical applications of John ellipsoids include outlier detection [ST80] and pattern recognition [Ros65, Gli98]. In the optimization literature, computing John ellipsoids is a fundamental ingredient for the ellipsoid method for linear programming [Kha79], cutting plane methods [TKE88], and convex programming [Sho77, Vai96]. Other applications in theoretical computer science include sampling [Vem05, CDWY18, GN23], bandits [BCK12, HK16], differential privacy [NTZ13], coresets [TMF20, TWZ+22], and randomized numerical linear algebra [Cla05, DDH+09, WY23, BMV23].

We refer to a survey of Todd [Tod16] for an account on algorithms and applications of John ellipsoids.

1.1 John ellipsoids via iterated leverage scores

Following a long line of work on algorithms for computing John ellipsoids [Wol70, Atw73, KT93, NN94, Kha96, STT78, KY05, SF04, TY07, AST08, Yu11, HFR20] based on convex optimization techniques, the work of [CCLY19] developed a new approach towards computing approximate John

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

ellipsoids via a simple fixed point iteration approach. The approach of [CCLY19] starts with an observation on the optimality condition for the dual problem, given by

$$\begin{aligned} &\text{minimize} \sum_{i=1}^{n} \mathbf{w}_i - \log\det(\mathbf{A}^\top \mathbf{W}\mathbf{A}) - d \\ &\text{subject to } \mathbf{w}_i \geq 0, i \in [n] \end{aligned}$$

where $\mathbf{W} = \mathrm{diag}(\mathbf{w})$.[1] The optimality condition requires that the dual weights $\mathbf{w}$ satisfy

$$\mathbf{w}_i = \boldsymbol{\tau}_i(\sqrt{\mathbf{W}}\mathbf{A}), \tag{1}$$

for each $i \in [n]$, where $\boldsymbol{\tau}_i(\mathbf{A})$ for a matrix $\mathbf{A}$ denotes the *i-th leverage score* of $\mathbf{A}$.

Definition 1.1 (Leverage score)**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$. Then, for each $i \in [n]$, we define the i-th leverage score as*

$$\boldsymbol{\tau}_i(\mathbf{A}) \coloneqq \mathbf{a}_i^\top (\mathbf{A}^\top \mathbf{A})^- \mathbf{a}_i = \sup_{\mathbf{A}\mathbf{x}\neq 0} \frac{[\mathbf{A}\mathbf{x}](i)^2}{\|\mathbf{A}\mathbf{x}\|_2^2}.$$

The optimality condition of (1) can be viewed as a fixed point condition, which suggests the following iterative algorithm for computing approximate John ellipsoids

$$\mathbf{w}_i^{(t)} \leftarrow \boldsymbol{\tau}_i(\sqrt{\mathbf{W}^{(t-1)}}\mathbf{A}) = \mathbf{w}_i^{(t-1)} \cdot \mathbf{a}_i^\top (\mathbf{A}^\top \mathbf{W}^{(t-1)}\mathbf{A})^- \mathbf{a}_i \tag{2}$$

where $\mathbf{W}^{(t)} \coloneqq \mathrm{diag}(\mathbf{w}^{(t)})$. After repeating this update for $T = O(\varepsilon^{-1}\log(n/d))$ iterations starting with $\mathbf{w}^{(0)} = d/n \cdot 1_n$, it can be shown that the ellipsoid Q defined by the quadratic form $\mathbf{Q} = \mathbf{A}^\top \mathbf{W}^{(T)}\mathbf{A}$, i.e. $Q = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x}^\top \mathbf{Q}\mathbf{x} \leq 1\}$, satisfies

$$\frac{1}{\sqrt{1+\varepsilon}} Q \subseteq P \subseteq \sqrt{d} \cdot Q. \tag{3}$$

Note that the computation of leverage scores can be done in $O(nd^{\omega-1})$ time, where $\omega \leq 2.371552$ is the current exponent of fast matrix multiplication [DWZ23, WXXZ24]. Thus, this gives an algorithm running in time $O(\varepsilon^{-1}nd^{\omega-1}\log(n/d))$ for outputting an ellipsoid with the guarantee of (3).

It is known that input matrices with additional structure admit even faster implementation of this algorithm. For instance, [CCLY19, SYYZ22] give faster algorithms for sparse matrices $\mathbf{A}$ for which the number of nonzero entries $\mathsf{nnz}(\mathbf{A})$ is much less than nd. The work of [SYYZ22] shows that this approach can also be sped up for matrices $\mathbf{A}$ with small treewidth.

1.2 Our results

Our first main result is a substantially faster algorithm for computing John ellipsoids. In the typical regime where $n \gg d$, our algorithm runs in $O(\varepsilon^{-1}nd)\log(n/d)$ time to output an ellipsoid Q satisfying (3), and $O(\varepsilon^{-1}nd^2)\log(n/d)$ time to output an ellipsoid Q which approximates the maximal volume up to a $(1+\varepsilon)$ factor. We will discuss our techniques for this result in Sections 1.2.1 and 1.2.2.

Table 1: Running time of John ellipsoid approximation for dense $n \times d$ matrices, for $n \gg d \gg \mathrm{poly}(\varepsilon^{-1}\log n)$. There is other prior work on sparse matrices and matrices with low treewidth [CCLY19, SYYZ22].

	Running time	Guarantee
[KY05, TY07]	$O(\varepsilon^{-1}nd^{\omega})$	volume approximation
[CCLY19]	$O(\varepsilon^{-1}nd^{\omega-1})\log(n/d)$	(3)
[CCLY19, SYYZ22]	$O(\varepsilon^{-2}nd)(\log n)\log(n/d)$	(3)
Theorem 1.6	$O(\varepsilon^{-1}nd)\log(n/d)$	(3)

[1] For a weight vector $\mathbf{w} \in \mathbb{R}^n$, we will often write the corresponding $n \times n$ diagonal matrix $\mathrm{diag}(\mathbf{w})$ as the capitalized version $\mathbf{W}$.

1.2.1 Linear time leverage scores via fast matrix multiplication

We start by showing how to approximate leverage scores up to $(1+\varepsilon)$ factors in $\tilde{O}(nd)$ time, which had not been observed before to the best of our knowledge. Note that if we compute exact leverage scores using fast matrix multiplication, then this takes time $O(nd^{\omega-1})$. Alternatively, sketching-based algorithms for approximate leverage scores are known, which gives the following running time for sparse matrices $\mathbf{A}$ with $\mathsf{nnz}(\mathbf{A})$ nonzero entries.

Theorem 1.2 ([SS11, DMMW12, CW13])**.** *There is an algorithm which, with probability at least $1-\delta$, outputs $\boldsymbol{\tau}'_i$ for $i \in [n]$ such that*

$$\boldsymbol{\tau}'_i = (1 \pm \varepsilon)\boldsymbol{\tau}_i(\mathbf{A})$$

and runs in time $O(\varepsilon^{-2}\,\mathsf{nnz}(\mathbf{A})\log(n/\delta)) + \mathrm{poly}(d\varepsilon^{-1}\log(n/\delta))$.

If the goal is to compute $(1+\varepsilon)$-approximate leverage scores for a dense $n \times d$ matrix, then we are not aware of a previous result which does this in a nearly linear $\tilde{O}(nd)$ time, which we now show:

Theorem 1.3. *There is an algorithm which, with probability at least $1-\delta$, outputs $\boldsymbol{\tau}'_i$ for $i \in [n]$ such that*

$$\boldsymbol{\tau}'_i = (1 \pm \varepsilon)\boldsymbol{\tau}_i(\mathbf{A})$$

in time $O(nd)\,\mathrm{poly}\log(\varepsilon^{-1}\log(n/\delta)) + O(n)\,\mathrm{poly}(\varepsilon^{-1}\log(n/\delta))$.

Our improvement comes from improving the running time analysis of a sketching-based algorithm of [DMMW12] by using fast rectangular matrix multiplication. We will need the following result on fast matrix multiplication:

Theorem 1.4 ([Cop82, Wil11, Wil24])**.** *There is a constant $\alpha \geq 0.1$ and an algorithm for multiplying a $m \times m$ and a $m \times m^{\alpha}$ matrix in $O(m^2\,\mathrm{poly}\log m)$ time, under the assumption that field operations can be carried out in $O(1)$ time.*

By applying the above result in blocks, we get the following version of this result for rectangular matrix multiplication.

Corollary 1.5. *There is a constant $\alpha \geq 0.1$ and an algorithm for multiplying a $n \times d$ for $n \geq d$ and a $d \times t$ matrix in $O(nd + nt^{1/\alpha+1})\,\mathrm{poly}\log t$ time, under the assumption that field operations can be carried out in $O(1)$ time.*

Proof. Let $m = t^{1/\alpha}$. If $d \leq m$, then matrix multiplication takes only $O(ndt) = O(nt^{1/\alpha+1})$ time, so assume that $d \geq m$. We partition the first matrix into an $O(n/m) \times O(d/m)$ block matrix with blocks of size $m \times m$ and the second into an $O(d/m) \times 1$ block matrix with blocks of size $m \times m^{\alpha}$. By Theorem 1.4, each block matrix multiplication requires $O(m^2\,\mathrm{poly}\log m)$ time, and we have $O(nd/m^2)$ of these to do, which gives the desired running time. □

That is, the above result shows that when multiplying an $n \times d$ matrix $\mathbf{A}$ with a $d \times t$ matrix for a much smaller t, then this multiplication can be done in roughly $O(nd)\,\mathrm{poly}\log t$ time. The work of [DMMW12] shows that the leverage scores of $\mathbf{A}$ can be written as the row norms of $\mathbf{AR}$ for a $d \times t$ matrix with $t = O(\varepsilon^{-2}\log(n/\delta))$, and thus this gives us the result of Theorem 1.3.

1.2.2 John ellipsoids via lazy updates

By using Theorem 1.3, we already obtain a John ellipsoid algorithm which runs in time $O(\varepsilon^{-1}nd)\log(n/d)\,\mathrm{poly}\log(\varepsilon^{-1}\log n)$ time, which substantially improves upon prior algorithms for dense input matrices $\mathbf{A}$. We now show how to obtain further improvements by using the idea of *lazy updates*. At the heart of our idea is to only compute the *quadratic forms* for the John ellipsoids for most iterations, and defer the computation of the weights until we have computed roughly $O(\log n)$ iterations. At the end of this group of iterations, we can then compute the John ellipsoid weights via fast matrix multiplication as used in Theorem 1.3, which allows us to remove the suboptimal $\mathrm{poly}\log\log n$ terms in the dominating term of the running time.

Theorem 1.6. *Given $\mathbf{A} \in \mathbb{R}^{n\times d}$, let P be the polytope defined by $P = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{Ax}\|_\infty \leq 1\}$. For $\varepsilon \in (0,1)$, there is an algorithm, Algorithm 3, that runs in time*

$$O(\varepsilon^{-1}nd)(\log(n/d) + \mathrm{poly}\log(\varepsilon^{-1}\log n)) + O(n)\,\mathrm{poly}(\varepsilon^{-1}\log n) + O(n^{0.1})d^{\omega+1}\varepsilon^{-3}(\log n)^2$$

and returns an ellipsoid Q such that $\frac{1}{\sqrt{1+\varepsilon}} \cdot Q \subseteq P \subseteq \sqrt{d} \cdot Q$ with probability at least $1 - 1/\mathrm{poly}(n)$.

The full proof of this result is given in Section 2.

Let $\mathbf{Q}^{(t)} = \mathbf{A}^\top \mathbf{W}^{(t)} \mathbf{A}$, where the weights $\mathbf{w}^{(t)}$ are defined as (2). Note that with this notation, the update rule for the iterative algorithm of [CCLY19] can be written as

$$\mathbf{w}_i^{(t)} = \mathbf{w}_i^{(t-1)} \cdot \mathbf{a}_i^\top (\mathbf{Q}^{(t-1)})^- \mathbf{a}_i. \tag{4}$$

Thus, given high-accuracy spectral estimates to the quadratics $\mathbf{Q}^{(t)}$, we can recover the weights $\mathbf{w}_i^{(t)}$ to high accuracy in $O(d^\omega)$ time per iteration by evaluating the quadratic forms $\mathbf{a}_i^\top (\mathbf{Q}^{(t)})^- \mathbf{a}_i$ and then multiplying them together. This approach is useful for fast algorithms if we only need to to do this for a small number of indices $i \in [n]$. This is indeed the case if we only need these weights for a *row sample* of $\sqrt{\mathbf{W}^{(t)}}\mathbf{A}$, which is sufficient for computing a spectral approximation to the next quadratic form $\mathbf{Q}^{(t)}$. Furthermore, we only need low-accuracy leverage scores (up to a factor of, say, $n^{0.1}$) to obtain a good row sample, which can be done quickly for all n rows [LMP13, CLM+15]. Thus, by repeatedly sampling rows of $\sqrt{\mathbf{W}^{(t)}}\mathbf{A}$, computing high-accuracy weights on the sampled rows, and then building an approximate quadratic, we can iteratively compute high-accuracy approximations $\tilde{\mathbf{Q}}^{(t)}$ to the quadratics $\mathbf{Q}^{(t)}$. More formally, our algorithm takes the following steps:

- We first compute low-accuracy leverage scores of $\sqrt{\mathbf{W}^{(t-1)}}\mathbf{A}$, which can be done in $O(nd)$ time. This gives us the weights $\mathbf{w}^{(t)}$ to low accuracy, say $\mathbf{u}^{(t)}$, for all n rows.
- We use the low-accuracy weights $\mathbf{u}^{(t)}$ to obtain a weighted subset of rows of $\sqrt{\mathbf{U}^{(t)}}\mathbf{A}$ which spectrally approximates $\mathbf{Q}^{(t)}$. Note, however, that we do not yet have the sampled rows of $\sqrt{\mathbf{W}^{(t)}}\mathbf{A}$ to high accuracy, since we do not know the weights $\mathbf{w}^{(t)}$ to high accuracy.
- If we only need the weights $\mathbf{w}^{(t)}$ for a small number of sampled rows $S \subseteq [n]$, then we can explicitly compute these using (4), since we inductively have access to high-accuracy quadratics $\tilde{\mathbf{Q}}^{(t')}$ for $t' = 0, 1, 2, \ldots, t-1$. These can then be used to build $\tilde{\mathbf{Q}}^{(t)}$.

While this algorithm allows us to quickly compute high-accuracy approximate quadratics $\tilde{\mathbf{Q}}^{(t)}$, this algorithm cannot be run for too many iterations, as the error in the low-accuracy leverage scores $\mathbf{u}^{(t)}$ grows to $\mathrm{poly}(n)$ factors in $O(\log n)$ rounds. This is a problem, as this error factor directly influences the number of leverage score samples needed to approximate $\tilde{\mathbf{Q}}^{(t)}$. We will now use the fast matrix multiplication trick from the previous Section 1.2.1 to fix this problem. Indeed, after $O(\log n)$ iterations, we will now have approximate quadratics $\tilde{\mathbf{Q}}^{(1)}, \tilde{\mathbf{Q}}^{(2)}, \ldots, \tilde{\mathbf{Q}}^{(t)}$ for $t = O(\log n)$. Now we just need to compute the n John ellipsoid weights which are given by

$$\mathbf{v}_i^{(t)} = \prod_{t'=1}^{t} \|\mathbf{e}_i^\top \mathbf{A} (\tilde{\mathbf{Q}}^{(t'-1)})^{-1/2}\|_2^2.$$

To approximate this quickly, we can approximate each term $\|\mathbf{e}_i^\top \mathbf{A} (\tilde{\mathbf{Q}}^{(t'-1)})^{-1/2}\|_2^2$ by $\|\mathbf{e}_i^\top \mathbf{A} (\tilde{\mathbf{Q}}^{(t'-1)})^{-1/2} \mathbf{G}^{(t')}\|_2^2$ for a random Gaussian matrix $\mathbf{G}^{(t')}$, by the Johnson–Lindenstrauss lemma [JL84]. Here, the number of columns of the Gaussian matrix can be taken to be $\mathrm{poly}(\varepsilon^{-1} \log n)$, so now all we need to compute is the matrix product

$$\mathbf{A} \cdot [(\tilde{\mathbf{Q}}^{(0)})^{-1/2} \mathbf{G}^{(0)}, (\tilde{\mathbf{Q}}^{(1)})^{-1/2} \mathbf{G}^{(1)}, \ldots, (\tilde{\mathbf{Q}}^{(t)})^{-1/2} \mathbf{G}^{(t)}]$$

which is the product of a $n \times d$ matrix and a $d \times m$ matrix for $m = \mathrm{poly}(\varepsilon^{-1} \log n)$. By Theorem 1.4, this can be computed in $O(nd \,\mathrm{poly} \log m)$ time. However, this resetting procedure is only run $O(\varepsilon^{-1})$ times across the $T = O(\varepsilon^{-1} \log n)$ iterations, so the running time contribution from the resetting is just

$$O(\varepsilon^{-1} nd) \,\mathrm{poly} \log(\varepsilon^{-1} \log n) + O(n) \,\mathrm{poly}(\varepsilon^{-1} \log n).$$

Overall, the total running time of our algorithm is

$$O(\varepsilon^{-1} nd)(\log(n/d) + \mathrm{poly} \log(\varepsilon^{-1} \log n)) + O(n) \,\mathrm{poly}(\varepsilon^{-1} \log n).$$

Remark 1.7. *In general, our techniques can be be viewed as a way of exploiting the increased efficiency of matrix multiplication when performed on a larger instance by delaying large matrix multiplication operations, so that the running time is $O(\varepsilon^{-1} nd \log(n/d)) + O(\varepsilon^{-1}) T_r$ where T_r is the time that it takes to multiply $\mathbf{A}$ by a $d \times r$ matrix for $r = \mathrm{poly}(\varepsilon^{-1} \log n)$. While we have instantiated this general theme by obtaining faster running times via fast matrix multiplication, one can expect similar improvements by other ways of exploiting the economies of scale of matrix multiplication, such as parallelization. For instance, we recover the same running time if we can multiply r vectors in parallel so that $T_r = O(nd)$.*

1.2.3 Streaming algorithms

The problem of computing John ellipsoids is also well-studied in the *streaming model*, where the input points $\mathbf{a}_i$ arrive one at a time in a stream [MSS10, AS15, WY22, MMO22, MMO23]. The streaming model is often considered when the number of points n is so large that we cannot fit all of the points in memory at once, and the focus is primarily on designing algorithms with low space complexity. Our second result of this work is that approaches similar to the one we take to prove Theorem 1.6 in fact also give a low-space implementation of the iterative John ellipsoid algorithm of [CCLY19].

Theorem 1.8 (Streaming algorithms)**.** *Given $\mathbf{A} \in \mathbb{R}^{n\times d}$, let P be the polytope defined by $P = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{A}\mathbf{x}\|_\infty \leq 1\}$. Furthermore, suppose that $\mathbf{A}$ is presented in a stream where the rows $\mathbf{a}_i \in \mathbb{R}^d$ arrive one by one in a stream. For $\varepsilon \in (0,1)$, there is an algorithm, Algorithm 1, that makes $T = O(\varepsilon^{-1}\log(n/d))$ passes over the stream, takes $O(d^2T)$ time to update after each new row, and returns an ellipsoid Q such that $\frac{1}{\sqrt{1+\varepsilon}}\cdot Q \subseteq P \subseteq \sqrt{d}\cdot Q$. Furthermore, the algorithm uses at most $O(d^2T)$ words of space.*

In Section 1.2.2, we showed that by storing only the quadratics $\tilde{\mathbf{Q}}^{(t)}$ and only computing the weights $\prod_{t'=1}^{t}\mathbf{a}_i(\tilde{\mathbf{Q}}^{(t'-1)})^-\mathbf{a}_i$ as needed, we could design fast algorithms for John ellipsoids. In fact, this idea is also useful in the streaming setting, since storing all of the weights $\mathbf{w}_i^{(t)}$ requires $O(n)$ space per iteration, whereas storing the quadratics $\tilde{\mathbf{Q}}^{(t)}$ requires only $O(d^2)$ space per iteration. Furthermore, in the streaming setting, we may optimize the update running time by instead storing the pseudo-inverse of the quadratics $(\tilde{\mathbf{Q}}^{(t)})^-$ and then updating them by using the Sherman-Morrison low rank update formula.[2] This gives the result of Theorem 1.8.

Algorithm 1 Streaming John ellipsoids via lazy updates

function StreamingJohnEllipsoid(input matrix $\mathbf{A}$)
  for $t = 0$ to T **do**
    Let $\mathbf{Q}^{(t)} = 0$
    for $i = 1$ to n **do**
      if $t = 0$ **then**
        Let $\mathbf{Q}^{(t)} \leftarrow \mathbf{Q}^{(t)} + \mathbf{a}_i\mathbf{a}_i^\top$
      else
        Let $\mathbf{w}_i^{(t)} = \prod_{t'=1}^{t}\mathbf{a}_i^\top(\mathbf{Q}^{(t'-1)})^-\mathbf{a}_i$
        Let $\mathbf{Q}^{(t)} \leftarrow \mathbf{Q}^{(t)} + \mathbf{w}_i^{(t)}\mathbf{a}_i\mathbf{a}_i^\top$
  return $\frac{1}{T+1}\sum_{t=0}^{T}\mathbf{Q}^{(t)}$

1.3 Notation

Throughout this paper, $\mathbf{A}$ will denote an $n\times d$ matrix whose n rows are given by vectors $\mathbf{a}_i \in \mathbb{R}^d$. For positive numbers $a, b > 0$, we write $a = (1\pm\varepsilon)b$ to mean that $(1-\varepsilon)b \leq a \leq (1+\varepsilon)b$. For symmetric positive semidefinite matrices $\mathbf{Q}, \mathbf{R}$, we write $\mathbf{Q} = (1\pm\varepsilon)\mathbf{R}$ to mean that $(1-\varepsilon)\mathbf{R} \preceq \mathbf{Q} \preceq (1+\varepsilon)\mathbf{R}$, where $\preceq$ denotes the Löwner order on PSD matrices.

2 Fast algorithms

2.1 Approximating the quadratics

We will analyze the following algorithm, Algorithm 2, for approximating the quadratics $\mathbf{Q}^{(t)}$ of the iterative John ellipsoid algorithm.

Fix a row i. Note then that for each iteration t, $\|\mathbf{e}_i^\top\mathbf{A}(\tilde{\mathbf{Q}}^{(t-1)})^{-1/2}\mathbf{G}\|_2^2$ is distributed as an independent χ^2 variable with k degrees of freedom, scaled by $\|\mathbf{e}_i^\top\mathbf{A}(\tilde{\mathbf{Q}}^{(t-1)})^{-1/2}\|_2^2$, say X_t. Note then that

[2] We note that storing the pseudo-inverse may increase the space complexity by polynomial factors in the bit complexity model.

Algorithm 2 Approximating the quadratics

function APPROXQUADRATIC(input matrix $\mathbf{A}$, initial weights $\tilde{\mathbf{w}}^{(0)}$, number of rounds T)
  Let $\tilde{\mathbf{Q}}^{(0)} = \mathbf{A}^\top \tilde{\mathbf{W}}^{(0)} \mathbf{A}$ for $\tilde{\mathbf{W}}^{(0)} = \mathrm{diag}(\tilde{\mathbf{w}}^{(0)})$.
  for $t = 1$ to T **do**
    Compute $\mathbf{A}(\tilde{\mathbf{Q}}^{(t-1)})^{-1/2}\mathbf{G}$ for a $d \times k$ Gaussian matrix $\mathbf{G}$.
    Let $\mathbf{u}_i^{(t)} = \mathbf{u}_i^{(t-1)} \cdot \|\mathbf{e}_i^\top \mathbf{A}(\tilde{\mathbf{Q}}^{(t-1)})^{-1/2}\mathbf{G}\|_2^2$ for each $i \in [n]$.
    Let $\mathbf{S}^{(t)}$ be a $(1+\varepsilon)$-approximate leverage score sample of $\sqrt{\mathbf{U}^{(t)}}\mathbf{A}$ (Thm. 2.3).
    For rows i sampled by $\mathbf{S}^{(t)}$, set $\mathbf{v}_i^{(t)} = \prod_{t'=1}^{t} \mathbf{a}_i^\top (\tilde{\mathbf{Q}}^{(t'-1)})^- \mathbf{a}_i$.
    Let $\tilde{\mathbf{Q}}^{(t)} = (\mathbf{S}^{(t)}\sqrt{\mathbf{V}^{(t)}}\mathbf{A})^\top \mathbf{S}^{(t)}\sqrt{\mathbf{V}^{(t)}}\mathbf{A}$.
  return $\{\tilde{\mathbf{Q}}^{(t)}\}_{t=0}^{T}$

after T iterations,

$$\mathbf{u}_i^{(T)} = \prod_{t=1}^{T} \|\mathbf{e}_i^\top \mathbf{A}(\tilde{\mathbf{Q}}^{(t-1)})^{-1/2}\mathbf{G}\|_2^2,$$

which is distributed as

$$\mathbf{u}_i^{(T)} \sim \prod_{t=1}^{T} \|\mathbf{e}_i^\top \mathbf{A}(\tilde{\mathbf{Q}}^{(t-1)})^{-1/2}\|_2^2 \cdot X_t = \prod_{t=1}^{T} \|\mathbf{e}_i^\top \mathbf{A}(\tilde{\mathbf{Q}}^{(t-1)})^{-1/2}\|_2^2 \cdot \prod_{t=1}^{T} X_t \tag{5}$$

for i.i.d. χ^2 variables X_t with k degrees of freedom. We will now bound each of the terms in this product.

2.1.1 Bounds on products of χ^2 variables

We need the following bound on products of χ^2 variables, which generalizes [SSK17, Proposition 1].

Lemma 2.1. *Let $X_1, X_2, \ldots, X_t$ be t i.i.d. χ^2 variables with k degrees of freedom. Then,*

$$\mathbf{Pr}\left\{\prod_{i=1}^{t} X_i \le \frac{1}{R}\right\} \le \inf_{s \in (0, k/2)} C_{-s,k}^t R^{-s}$$

and

$$\mathbf{Pr}\left\{\prod_{i=1}^{t} X_i \ge R\right\} \le \inf_{s > -k/2} C_{s,k}^t R^{-s}$$

where

$$C_{s,k} = \frac{2^s \Gamma(s + k/2)}{\Gamma(k/2)} > 0$$

Proof. For $s > -k/2$, the moment generating function of $\log X_i$ is given by

$$\mathbf{E}\, e^{s \log X_i} = \mathbf{E}\, X_i^s = \frac{1}{2^{k/2}\Gamma(k/2)} \int_0^\infty x^s x^{k/2-1} e^{-x/2}\, dx = \frac{2^s \Gamma(s+k/2)}{\Gamma(k/2)} = C_{s,k}.$$

Then by Chernoff bounds,

$$\mathbf{Pr}\left\{\prod_{i=1}^{t} X_i \le \frac{1}{R}\right\} = \mathbf{Pr}\left\{\exp\left(\sum_{i=1}^{t} -s \log X_i\right) \ge R^s\right\} \le \mathbf{E}[e^{-s \log X_i}]^t R^{-s} = C_{-s,k}^t R^{-s}$$

and

$$\mathbf{Pr}\left\{\prod_{i=1}^{t} X_i \ge R\right\} = \mathbf{Pr}\left\{\exp\left(\sum_{i=1}^{t} s \log X_i\right) \ge R^s\right\} \le \mathbf{E}[e^{s \log X_i}]^t R^{-s} = C_{s,k}^t R^{-s}$$

□

Using the above result, we can show that if the χ^2 variables have $k = O(1/\theta)$ degrees of freedom and the number of rounds T is $c \log n$ for a small enough constant c, then the product of the χ^2 variables $\prod_{t=1}^{T} X_t$ will be within a n^θ factor for some small constant $\theta > 0$.

Lemma 2.2. *Fix a small constant $\theta > 0$ and let $k = O(1/\theta)$. Let $T = c \log n$ for a sufficiently small constant $c > 0$. Let $X_1, X_2, \ldots, X_T$ be t i.i.d. χ^2 variables with k degrees of freedom. Then,*

$$\mathbf{Pr}\left\{ \frac{1}{n^\theta} \leq \prod_{i=1}^{t} X_i \leq n^\theta \right\} \geq 1 - \frac{1}{\mathrm{poly}(n)}.$$

Proof. Set $s = k/2$. Then, $C_{-s,k}$ and $C_{s,k}$ are absolute constants. Now let c to be small enough (depending on s and k) such that for $T = c \log n$, $C_{-s,k}^T, C_{s,k}^T \leq n$. Furthermore, set $R = n^\theta$. Then, by Lemma 2.1, we have that both $\mathbf{Pr}\{\prod_{i=1}^{t} X_i \leq \frac{1}{R}\}$ and $\mathbf{Pr}\{\prod_{i=1}^{t} X_i \geq R\}$ are bounded by $n \cdot R^{-s} = n \cdot n^{\theta \cdot s} = 1/\mathrm{poly}(n)$, as desired. □

It follows that with probability at least $1 - 1/\mathrm{poly}(n)$, the products of χ^2 variables appearing in (5) are bounded by n^θ for every row $i \in [n]$ and for every iteration $t \in [T]$. We will condition on this event in the following discussion.

2.1.2 Bounds on the quadratic $\tilde{\mathbf{Q}}^{(t)}$

In the previous section, we have established that the products of χ^2 variables in (5) are bounded by n^θ. We will now use this fact to show that the quadratics $\tilde{\mathbf{Q}}^{(t-1)}$ in Algorithm 2 are good approximate John ellipsoid quadratics. We will use the following leverage score sampling theorem for this.

Theorem 2.3 ([DMM06, SS11])**.** *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$. Let $\boldsymbol{\tau}'_i \geq \boldsymbol{\tau}_i(\mathbf{A})$ and let $p_i = \min\{1, \boldsymbol{\tau}'_i/\alpha\}$ for $\alpha = \Theta(\varepsilon^2)/\log(d/\delta)$. If $\mathbf{S}$ is a diagonal matrix with $\mathbf{S}_{i,i} = 1/\sqrt{p_i}$ with probability p_i and 0 otherwise for $i \in [n]$, then with probability at least $1 - \delta$,*

$$\text{for all } \mathbf{x} \in \mathbb{R}^d, \qquad \|\mathbf{SAx}\|_2^2 = (1 \pm \varepsilon)\|\mathbf{Ax}\|_2^2.$$

This theorem, combined with the bounds on χ^2 products, gives the following guarantee for the approximate quadratics $\tilde{\mathbf{Q}}^{(t)}$.

Lemma 2.4. *Fix a small constant $\theta > 0$ and let $k = O(1/\theta)$. Suppose that the leverage score sample in Line 6 oversamples by a factor of $O(n^{2\theta})$, that is, uses leverage score estimates $\boldsymbol{\tau}'_i$ such that $\boldsymbol{\tau}'_i \geq O(n^{2\theta})\boldsymbol{\tau}_i(\sqrt{\mathbf{U}^{(t)}}\mathbf{A})$. Then, with probability at least $1 - 1/\mathrm{poly}(n)$, we have for every $t \in [T]$ that*

$$\tilde{\mathbf{Q}}^{(t)} = (1 \pm \varepsilon)\mathbf{A}^\top \mathbf{V}^{(t)} \mathbf{A} \tag{6}$$

where $\mathbf{v}_i^{(t)} = \prod_{t'=1}^{t-1} \mathbf{a}_i^\top (\tilde{\mathbf{Q}}^{(t')})^- \mathbf{a}_i$ for $i \in [n]$. Furthermore, $\tilde{\mathbf{Q}}^{(t)}$ can be computed in $O(n^{2\theta}) T \varepsilon^{-2} d^{\omega+1} \log n$ time in each iteration.

Proof. We will first condition on the success of the event of Lemma 2.2, so that the χ^2 products in (5) are bounded by n^θ factors for every row $i \in [n]$ and every iteration $t \in [t]$. We will also condition on the success of the leverage score sampling for all T iterations.

Note that by (5) and the bound the χ^2 products, $\mathbf{u}_i^{(t)}$ is within a $O(n^{2\theta})$ factor of $\mathbf{v}_i^{(t)}$, and thus $\mathbf{S}^{(t)}$ is a correct leverage score sample for $\sqrt{\mathbf{V}^{(t)}}\mathbf{A}$. We thus have that

$$\tilde{\mathbf{Q}}^{(t)} = (\mathbf{S}^{(t)}\sqrt{\mathbf{V}^{(t)}}\mathbf{A})^\top \mathbf{S}^{(t)}\sqrt{\mathbf{V}^{(t)}}\mathbf{A} = (1 \pm \varepsilon)\mathbf{A}^\top \mathbf{V}^{(t)} \mathbf{A}.$$

For the running time, note that $\mathbf{S}^{(t)}$ samples at most $O(\varepsilon^{-2} n^{2\theta} d \log n)$ rows in each iteration, and each sampled row i requires $O(d^\omega T)$ to compute $\mathbf{v}_i^{(t)}$. This gives the running time claim. □

Algorithm 3 John ellipsoids via lazy updates

function JOHNELLIPSOID(input matrix $\mathbf{A}$)
  Let $B = O(c^{-1}\varepsilon^{-1})$ and $T = O(c\log(n/d))$ for a sufficiently small constant c.
  Let $\tilde{\mathbf{w}}_i^{(0)} = d/n$ for $i \in [n]$.
  for $b = 1$ to B **do**
    Let $\{\tilde{\mathbf{Q}}^{(t)}\}_{t=0}^{T}$ be given by APPROXQUADRATIC$(\mathbf{A}, \tilde{\mathbf{w}}^{(0)})$ (Algorithm 2).
    Let $\mathbf{G}^{(t-1)}$ for $t \in [T]$ be a random $d \times m$ Gaussian for $m = O(\varepsilon^{-2}(BT)^2 \log n)$.
    Compute $\mathbf{A} \cdot [(\tilde{\mathbf{Q}}^{(0)})^{-1/2}\mathbf{G}^{(0)}, (\tilde{\mathbf{Q}}^{(1)})^{-1/2}\mathbf{G}^{(1)}, \ldots, (\tilde{\mathbf{Q}}^{(T)})^{-1/2}\mathbf{G}^{(T)}]$.
    Let $\tilde{\mathbf{w}}_i^{(b,t)} = \prod_{t'=1}^{t} \frac{1}{m}\|\mathbf{e}_i^\top \mathbf{A}(\tilde{\mathbf{Q}}^{(t'-1)})^{-1/2}\mathbf{G}^{(t'-1)}\|_2^2$ for each $i \in [n]$.
    Let $\tilde{\mathbf{w}}^{(0)} = \tilde{\mathbf{w}}^{(b,T)}$.
  Let $\tilde{\mathbf{w}} = \frac{1}{BT}\sum_{b,t} \tilde{\mathbf{w}}^{(b,t)}$.
  return $\mathbf{A}^\top \tilde{\mathbf{W}} \mathbf{A}$

2.2 Full algorithm

We now combine the subroutine for approximating quadratics from Section 2.1 with a resetting procedure using fast matrix multiplication to obtain our full algorithm for quickly computing John ellipsoids. The full algorithm is presented in Algorithm 3.

We will need the following theorem, which summarizes results of [CCLY19] on guarantees of the fixed point iteration algorithm (2) under approximate leverage score computations.

Theorem 2.5 (Lemma 2.3 and Lemma C.4 of [CCLY19]). *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ and let $P = \{\mathbf{x} : \|\mathbf{A}\mathbf{x}\|_\infty \le 1\}$. Let $T = O(\varepsilon^{-1}\log(n/d))$. Suppose that*

$$\mathbf{w}_i^{(t)} = (1 \pm \varepsilon)\boldsymbol{\tau}_i(\sqrt{\mathbf{W}^{(t-1)}}\mathbf{A})$$

for all $t \in [T]$ and $i \in [n]$. Then, if Q is the ellipsoid given by the quadratic form $\mathbf{A}^\top \mathbf{W}^{(T)}\mathbf{A}$, then $\frac{1}{\sqrt{1+\varepsilon}} \cdot Q \subseteq P \subseteq \sqrt{d} \cdot Q$.

We also use the Johnson–Lindenstrauss lemma, which is a standard tool for randomized numerical linear algebra, especially in the context of approximating leverage scores [SS11, DMMW12, LMP13, CLM+15].

Theorem 2.6 ([JL84, DG03]). *Let $m = O(\varepsilon^{-2}\log(n/\delta))$ and let $\mathbf{G}$ be a random $m \times d$ Gaussian matrix. Then, for any n points $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n \in \mathbb{R}^d$ in d dimensions, we have with probability at least $1 - \delta$ that*

$$\frac{1}{m}\|\mathbf{G}\mathbf{a}_i\|_2^2 = (1 \pm \varepsilon)\|\mathbf{a}_i\|_2^2$$

simultaneously for every $i \in [n]$.

We will now give a proof of Theorem 1.6.

Proof of Theorem 1.6. We first argue the correctness of this algorithm. By Theorem 2.5, it suffices to verify that $\tilde{\mathbf{w}}_i^{(b,t)}$ computes $(1+\varepsilon)$-approximate leverage scores in order to prove the claimed guarantees of Theorem 1.6. For the updates within APPROXQUADRATIC (Algorithm 2), we have by Lemma 2.4 that

$$\tilde{\mathbf{Q}}^{(t)} = (1 \pm \varepsilon)\mathbf{A}^\top \mathbf{V}^{(t)}\mathbf{A}.$$

Thus,

$$\begin{aligned}\mathbf{v}_i^{(t-1)} \cdot \|\mathbf{e}_i^\top \mathbf{A}(\tilde{\mathbf{Q}}^{(t-1)})^{-1/2}\|_2^2 &= (1 \pm \varepsilon)\mathbf{v}_i^{(t-1)} \cdot \|\mathbf{e}_i^\top \mathbf{A}(\mathbf{A}^\top \mathbf{V}^{(t-1)}\mathbf{A})^{-1/2}\|_2^2 \\ &= (1 \pm \varepsilon)\boldsymbol{\tau}_i(\sqrt{\mathbf{V}^{(t-1)}}\mathbf{A})\end{aligned}$$

and thus the weights $\mathbf{v}_i^{(t)}$ output by APPROXQUADRATIC (Algorithm 2) indeed satisfy the requirements of Theorem 2.5.

For the weights computed in Line 8 of Algorithm 3, note that we also compute these weights, but this time with approximation error from the application of the Gaussian matrix $\frac{1}{m}\mathbf{G}^{(t)}$ to speed up the

computation. Applying Theorem 2.6 with ε set to ε/BT and failure probability $\delta = 1/\operatorname{poly}(n)$, we have that

$$\frac{1}{m}\|\mathbf{e}_i^\top \mathbf{A}(\tilde{\mathbf{Q}}^{(t'-1)})^{-1/2}\mathbf{G}^{(t'-1)}\|_2^2 = (1 \pm \varepsilon/BT)\|\mathbf{e}_i^\top \mathbf{A}(\tilde{\mathbf{Q}}^{(t'-1)})^{-1/2}\|_2^2.$$

Note then that Line 8 takes a product of at most BT of these approximations, so the total error in the approximation is at most

$$(1 \pm \varepsilon/BT)^{BT} = (1 \pm O(\varepsilon)).$$

The running time is given by BT iterations of the inner loop of ApproxQuadratic and B iterations of the fast matrix multiplication procedure in Line 7 of Algorithm 3. The inner loop of ApproxQuadratic requires $O(nd)$ time to compute the product with the $d \times k$ Gaussian as well as the time to compute the approximate quadratic, which is bounded in Lemma 2.4. Altogether, this gives the claimed running time bound. □

3 Future directions

In this work, we developed fast algorithms and low-space streaming algorithms for the problem of computing John ellipsoids. Our fast algorithms use a combination of using lazy updates together with fast matrix multiplication to substantially improve the running time of John ellipsoids, and we apply similar ideas to obtain a low-space streaming implementation of the John ellipsoid algorithm.

Our results have several limitations that we discuss here, which we leave for future work to resolve. First, our algorithm makes crucial use of fast matrix multiplication in order to get running time improvements. However, this makes it a less attractive option for practical implementations, and also makes the polynomial dependence on ε in the $\tilde{O}(n)\operatorname{poly}(\varepsilon^{-1})$ term rather large. Thus, it is an interesting question whether the running time that we obtain in Theorem 1.6 is possible without fast matrix multiplication.

Question 3.1. *Is there an algorithm with the guarantee of Theorem 1.6 that avoids fast matrix multiplication?*

More generally, it is an interesting question to design algorithms for approximating John ellipsoids with optimal running time. This is an old question which has been studied in a long line of work [Wol70, Atw73, KT93, NN94, Kha96, STT78, KY05, SF04, TY07, AST08, Yu11, HFR20], and we believe that the investigation of this question will lead to further interesting developments in algorithms research.

Question 3.2. *What is the optimal running time of approximating John ellipsoids?*

For instance, one interesting question is whether it is possible to obtain *nearly linear time* algorithms that run in time $\tilde{O}(nd) + \tilde{O}(n)\operatorname{poly}(\varepsilon^{-1})$, or even *input sparsity time algorithms* that run in time $\tilde{O}(\mathsf{nnz}(\mathbf{A}))+\tilde{O}(n)\operatorname{poly}(\varepsilon^{-1})$. The resolution of such questions for the least squares linear regression problem has led to great progress in algorithms, and studying these questions for the John ellipsoid problem may have interesting consequences as well.

John ellipsoids are closely related to ℓ_p *Lewis weights*, which give a natural ℓ_p generalization of John ellipsoids and leverage scores and have been a valuable tool in randomized numerical linear algebra. There has been a recent focus on fast algorithms for computing $(1+\varepsilon)$-approximate ℓ_p Lewis weights [FLPS22, AGS24], and thus it is natural to ask whether developments in algorithms for John ellipsoids would carry over to algorithms for ℓ_p Lewis weights as well.

Question 3.3. *What is the optimal running time of approximating ℓ_p Lewis weights?*

Finally, we raise questions concerning streaming algorithms for approximating John ellipsoids. In Theorem 1.8, we gave a multi-pass streaming algorithm which obtains a $(1+\varepsilon)$-optimal John ellipsoid. A natural question is whether a similar algorithm can be achieved in fewer passes, or if there are pass lower bounds for computing $(1+\varepsilon)$-optimal John ellipsoids.

Question 3.4. *Are there small space streaming algorithms for approximating John ellipsoids up to a $(1+\varepsilon)$ factor which make fewer than $O(\varepsilon^{-1}\log(n/d))$ passes, or do small space streaming algorithms necessarily require many passes?*

We note that there are one-pass small space streaming algorithms if we allow for approximation factors that scale as $O(\sqrt{\log n})$ rather than $(1+\varepsilon)$ [WY22, MMO22, MMO23].

Acknowledgments and Disclosure of Funding

The authors were supported in part by a Simons Investigator Award and NSF CCF-2335412.

Bibliography

[AGS24] Simon Apers, Sander Gribling, and Aaron Sidford. On computing approximate Lewis weights. *CoRR*, abs/2404.02881, 2024. 3

[AS15] Pankaj K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. *Algorithmica*, 72(1):83–98, 2015. 1.2.3

[AST08] Selin Damla Ahipasaoglu, Peng Sun, and Michael J. Todd. Linear convergence of a modified Frank-Wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optim. Methods Softw.*, 23(1):5–19, 2008. 1.1, 3

[Atw69] Corwin L. Atwood. Optimal and efficient designs of experiments. *Ann. Math. Statist.*, 40:1570–1602, 1969. 1

[Atw73] Corwin L. Atwood. Sequences converging to D-optimal designs of experiments. *Ann. Statist.*, 1:342–352, 1973. 1.1, 3

[BCK12] Sébastien Bubeck, Nicolò Cesa-Bianchi, and Sham M. Kakade. Towards minimax policies for online linear optimization with bandit feedback. In Shie Mannor, Nathan Srebro, and Robert C. Williamson, editors, *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, volume 23 of *JMLR Proceedings*, pages 41.1–41.14. JMLR.org, 2012. 1

[BMV23] Aditya Bhaskara, Sepideh Mahabadi, and Ali Vakilian. Tight bounds for volumetric spanners and applications. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. 1

[CCLY19] Michael B. Cohen, Ben Cousins, Yin Tat Lee, and Xin Yang. A near-optimal algorithm for approximating the John ellipsoid. In Alina Beygelzimer and Daniel Hsu, editors, *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pages 849–873. PMLR, 2019. (document), 1.1, 1, 1, 1.2.2, 1.2.3, 2.2, 2.5

[CDWY18] Yuansi Chen, Raaz Dwivedi, Martin J. Wainwright, and Bin Yu. Fast MCMC sampling algorithms on polytopes. *J. Mach. Learn. Res.*, 19:55:1–55:86, 2018. 1

[Cla05] Kenneth L. Clarkson. Subgradient and sampling algorithms for ℓ_1 regression. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 257–266, USA, 2005. Society for Industrial and Applied Mathematics. 1

[CLM+15] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 181–190. ACM, 2015. 1.2.2, 2.2

[Cop82] Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. 1.4

[CW13] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 81–90. ACM, 2013. 1.2

[DDH+09] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM J. Comput.*, 38(5):2060–2078, 2009. 1

[DG03] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003. 2.6

[DMM06] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 1127–1136. ACM Press, 2006. 2.3

[DMMW12] Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *J. Mach. Learn. Res.*, 13:3475–3506, 2012. 1.2, 1.2.1, 1.2.1, 2.2

[DWZ23] Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 2129–2138. IEEE, 2023. 1

[FLPS22] Maryam Fazel, Yin Tat Lee, Swati Padmanabhan, and Aaron Sidford. Computing Lewis weights to high precision. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2723–2742. SIAM, 2022. 3

[Gli98] François Glineur. Pattern separation via ellipsoids and conic programming. *Mémoire de DEA, Faculté Polytechnique de Mons, Mons, Belgium*, 1998. 1

[GN23] Adam Gustafson and Hariharan Narayanan. John's walk. *Adv. in Appl. Probab.*, 55(2):473–491, 2023. 1

[HFR20] Radoslav Harman, Lenka Filová, and Peter Richtárik. A randomized exchange algorithm for computing optimal approximate designs of experiments. *J. Amer. Statist. Assoc.*, 115(529):348–361, 2020. 1.1, 3

[HK16] Elad Hazan and Zohar S. Karnin. Volumetric spanners: An efficient exploration basis for learning. *J. Mach. Learn. Res.*, 17:119:1–119:34, 2016. 1

[JL84] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemp. Math.*, pages 189–206. Amer. Math. Soc., Providence, RI, 1984. 1.2.2, 2.6

[Joh48] Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948*, pages 187–204. Interscience Publishers, Inc., New York, N. Y., 1948. 1

[Kha79] Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244(5):1093–1096, 1979. 1

[Kha96] Leonid G. Khachiyan. Rounding of polytopes in the real number model of computation. *Math. Oper. Res.*, 21(2):307–320, 1996. 1.1, 3

[KT93] Leonid G. Khachiyan and Michael J. Todd. On the complexity of approximating the maximal inscribed ellipsoid for a polytope. *Math. Programming*, 61(2, Ser. A):137–159, 1993. 1.1, 3

[KY05] Piyush Kumar and E. Alper Yildirim. Minimum-volume enclosing ellipsoids and core sets. *J. Optim. Theory Appl.*, 126(1):1–21, 2005. 1.1, 1, 3

[LMP13] Mu Li, Gary L. Miller, and Richard Peng. Iterative row sampling. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 127–136. IEEE Computer Society, 2013. 1.2.2, 2.2

[MMO22] Yury Makarychev, Naren Sarayu Manoj, and Max Ovsiankin. Streaming algorithms for ellipsoidal approximation of convex polytopes. In Po-Ling Loh and Maxim Raginsky, editors, *Conference on Learning Theory, 2-5 July 2022, London, UK*, volume 178 of *Proceedings of Machine Learning Research*, pages 3070–3093. PMLR, 2022. 1.2.3, 3

[MMO23] Yury Makarychev, Naren Sarayu Manoj, and Max Ovsiankin. Near-optimal streaming ellipsoidal rounding for general convex polytopes. *CoRR*, abs/2311.09460, 2023. 1.2.3, 3

[MSS10] Asish Mukhopadhyay, Animesh Sarker, and Tom Switzer. Approximate ellipsoid in the streaming model. In Weili Wu and Ovidiu Daescu, editors, *Combinatorial Optimization and Applications - 4th International Conference, COCOA 2010, Kailua-Kona, HI, USA, December 18-20, 2010, Proceedings, Part II*, volume 6509 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 2010. 1.2.3

[NN94] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994. 1.1, 3

[NTZ13] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 351–360. ACM, 2013. 1

[Ros65] Judah Ben Rosen. Pattern separation by convex programming. *Journal of Mathematical Analysis and Applications*, 10(1):123–134, 1965. 1

[SF04] Peng Sun and Robert M. Freund. Computation of minimum-volume covering ellipsoids. *Oper. Res.*, 52(5):690–706, 2004. 1.1, 3

[Sho77] Naum Z Shor. Cut-off method with space extension in convex programming problems. *Cybernetics*, 13(1):94–96, 1977. 1

[Sil13] Samuel Silvey. *Optimal design: an introduction to the theory for parameter estimation*, volume 1. Springer Science & Business Media, 2013. 1

[SS11] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011. 1.2, 2.3, 2.2

[SSK17] Željka Stojanac, Daniel Suess, and Martin Kliesch. On products of gaussian random variables. *arXiv preprint arXiv:1711.10516*, 2017. 2.1.1

[ST80] BW Silverman and DM Titterington. Minimum covering ellipses. *SIAM Journal on Scientific and Statistical Computing*, 1(4):401–409, 1980. 1

[STT78] Samuel D Silvey, DH Titterington, and Ben Torsney. An algorithm for optimal designs on a design space. *Communications in Statistics-Theory and Methods*, 7(14):1379–1389, 1978. 1.1, 3

[SYYZ22] Zhao Song, Xin Yang, Yuanyuan Yang, and Tianyi Zhou. Faster algorithm for structured John ellipsoid computation. *CoRR*, abs/2211.14407, 2022. 1, 1

[TKE88] S. P. Tarasov, L. G. Khachiyan, and I. I. Èrlikh. The method of inscribed ellipsoids. *Dokl. Akad. Nauk SSSR*, 298(5):1081–1085, 1988. 1

[TMF20] Murad Tukan, Alaa Maalouf, and Dan Feldman. Coresets for near-convex functions. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 1

[Tod16] Michael J. Todd. *Minimum volume ellipsoids - theory and algorithms*, volume 23 of *MOS-SIAM Series on Optimization*. SIAM, 2016. 1

[TWZ+22] Murad Tukan, Xuan Wu, Samson Zhou, Vladimir Braverman, and Dan Feldman. New coresets for projective clustering and applications. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pages 5391–5415. PMLR, 2022. 1

[TY07] Michael J. Todd and E. Alper Yildirim. On Khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids. *Discrete Appl. Math.*, 155(13):1731–1744, 2007. 1.1, 1, 3

[Vai96] Pravin M. Vaidya. A new algorithm for minimizing convex functions over convex sets. *Math. Programming*, 73(3, Ser. A):291–341, 1996. 1

[Vem05] Santosh Vempala. Geometric random walks: a survey. *Combinatorial and computational geometry*, 52(573-612):2, 2005. 1

[Wil11] Ryan Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 115–125. IEEE Computer Society, 2011. 1.4

[Wil24] Ryan Williams. Personal communication, 2024. 1.4

[Wol70] P. Wolfe. Convergence theory in nonlinear programming. In *Integer and nonlinear programming*, pages 1–36. North-Holland, Amsterdam-London, 1970. 1.1, 3

[WXXZ24] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 3792–3835. SIAM, 2024. 1

[WY22] David P. Woodruff and Taisuke Yasuda. High-dimensional geometric streaming in polynomial space. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 732–743. IEEE, 2022. 1.2.3, 3

[WY23] David P. Woodruff and Taisuke Yasuda. New subset selection algorithms for low rank approximation: Offline and online. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1802–1813. ACM, 2023. 1

[Yu11] Yaming Yu. D-optimal designs via a cocktail algorithm. *Stat. Comput.*, 21(4):475–481, 2011. 1.1, 3

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of our work in Section 3, and pose open questions towards closing these gaps.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide complete proofs of all of our results, and references to prior work whenever we need prior results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [NA]

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [NA]

Justification: **[TODO]**

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [NA]

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [NA]

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: **[TODO]**

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: **[TODO]**

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: **[TODO]**

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.