Supplementary Material for Leveraging Well-Conditioned Bases: Streaming and Distributed Summaries in Minkowski p-Norms

A. Proofs for Section 3

Lemma A.1. Denote the ith global leverage score of A by w_i and its associated local leverage score in a block of input A be denoted $\hat{w}_k$. Then $w_i/\text{poly}(d) \leq \hat{w}_k$. In particular, $w_i/(d\alpha^p\beta) \leq \hat{w}_k$.

Proof of Lemma A.1. Let $U = AR$. Recall that $w_i = \|\mathbf{e}_i^T AR\|_p^p$. Then for some coordinate j we must have $|\mathbf{e}_i^T AR\mathbf{e}_j|^p \geq w_i/d$. Taking $\mathbf{x} = \mathbf{e}_j$ we see that

$$|(AR\mathbf{x})_i|^p \geq \frac{w_i}{d}. \tag{2}$$

However, Fact 1 implies :

$$\|AR\mathbf{x}\|_p^p \leq \|AR\|_p^p \leq \alpha^p \leq \text{poly}(d). \tag{3}$$

Hence, there exists a $\mathbf{y} \in \text{col}(A)$ with $\mathbf{y} = AR\mathbf{x}$ such that $|\mathbf{y}_i|^p \geq w_i/d$ from Equation (2). Also, $\|\mathbf{y}\|_p^p \leq \alpha^p$ from Equation (3). Thus,

$$\frac{|\mathbf{y}_i|^p}{\|\mathbf{y}\|_p^p} \geq \frac{|\mathbf{y}_i|^p}{\alpha^p} \tag{4}$$

$$\geq \frac{w_i}{d\alpha^p} \tag{5}$$

$$\geq \frac{w_i}{\text{poly}(d)}. \tag{6}$$

From this we see;

$$w_i \leq \frac{|\mathbf{y}_i|^p d\alpha^p}{\|\mathbf{y}\|_p^p}. \tag{7}$$

Now, let B be a block of rows from A. We manipulate B by considering it either as an individual matrix or as a coordinate subspace of A; i.e all rows are zero except for those contained in B which will be denoted by $\hat{A}$. Define $\hat{\mathbf{y}} = \hat{A}R\mathbf{x}$. Then $\hat{\mathbf{y}}_{j'} = \mathbf{y}_{j'}$ when j' is a row from B and $\hat{\mathbf{y}}_{j'} = 0$ otherwise. Thus, $\|\hat{\mathbf{y}}\|_p^p \leq \|\mathbf{y}\|_p^p$ and:

$$w_i \leq \frac{|\mathbf{y}_i|^p d\alpha^p}{\|\hat{\mathbf{y}}\|_p^p}. \tag{8}$$

For rows i which are also found in B (indexed as k) we see that $|\hat{\mathbf{y}}_k|^p = |\mathbf{y}_i|^p$. So, *for such indices*, using Equations (7) and (8):

$$w_i \leq \frac{|\hat{\mathbf{y}}_k|^p d\alpha^p}{\|\hat{\mathbf{y}}\|_p^p}. \tag{9}$$

Since $\hat{\mathbf{y}}$ is the restriction of $\mathbf{y}$ to coordinates of B we can write $\hat{\mathbf{y}} = B\hat{R}\hat{x}$ where $B\hat{R}$ is well-conditioned. Let $\hat{w}_k = \|\mathbf{e}_k^T B\hat{R}\|_p^p$ be the kth local leverage score in B. By applying the same argument as in Fact 2 it can be shown that $|\hat{\mathbf{y}}_k|^p/\|\hat{\mathbf{y}}\|_p^p \leq \text{poly}(d)\hat{w}_k$. Indeed,

$$|\hat{\mathbf{y}}_k|^p = |(\hat{A}\hat{R}\hat{x})_k|^p \tag{10}$$

$$\leq \|e_k^T \hat{A}\hat{R}\|_p^p \|\hat{x}\|_q^p \quad \text{by Hölder's inequality} \tag{11}$$

$$\leq \hat{w}_k\beta\|B\hat{R}\hat{x}\|_p^p \tag{12}$$

$$\leq \beta\hat{w}_k\|\hat{\mathbf{y}}\|_p^p. \tag{13}$$

The second inequality uses condition 2 from Theorem 2.1 and the fact that $B\hat{R}$ is a well-conditioned basis. Then using Equation 9, the following then proves the latter claim of the lemma:

$$\frac{w_i}{d\alpha^p} \leq \frac{|\hat{\mathbf{y}}_k|^p}{\|\hat{\mathbf{y}}\|_p^p} \leq \beta\hat{w}_k.$$

Finally, Theorem 2.2 states that β is at most $\text{poly}(d)$ which proves the result. □

Lemma A.2. All global leverage scores above a threshold can be found by computing local leverage scores and increasing the space complexity by a $\text{poly}(d)$ factor.

Proof of Lemma A.2. First we determine the space necessary to find all leverage scores exceeding δ. Let $I = \{i : w_i > \delta\}$. Then $\alpha^p \geq \sum_{i=1}^n w_i \geq \sum_{i\in I} w_i \geq \delta|I|$ by arguing as in Fact 1 Hence, the space necessary is $|I| \leq \alpha^p/\delta$. Now focus on finding these rows in the streaming fashion. By Lemma A.1 we see that for rows k in the block which is stored from the stream we have the property that $w_i/d\alpha^p\beta \leq \hat{w}_k$. Hence, any $w_i > \delta$ results in $\hat{w}_k > \delta/d\alpha^p\beta$ for the local thresholding. So to keep all such $w_i > \delta$, we must store all $\hat{w}_k > \delta/d\alpha^p\beta = \hat{\delta}$. Arguing similarly as in Fact 1 again define $\hat{I} = \{k : \hat{w}_k > \hat{\delta}\}$ so that: $\alpha^p \geq \sum_k \hat{w}_k \geq \sum_{k\in\hat{I}} \hat{w}_k \geq \hat{\delta}|\hat{I}|$. Hence $|\hat{I}| \leq \alpha^p/\hat{\delta} = d\alpha^{2p}\beta/\delta$. That is, $|\hat{I}| \leq d\beta\alpha^p \cdot |I|$ which proves the claim as Theorem 2.2 states that all of the parameters are $\text{poly}(d)$. □

Proof of Theorem 3.3. We claim that the output of Algorithm 1 is a matrix B which contains rows of high leverage in A. The algorithm initially reads in b rows and inserts these to matrix A'. A well-conditioned basis U for A' is then computed using Theorem 2.2 and incurs the associated $O(bd^2 + bd^5 \log b)$ time. The matrix U and A' are passed to Algorithm 2 whereby if a row i in U has local leverage exceeding τ then row i of A' is kept. There are at most $\text{poly}(d)/\tau$ of these rows as seen in Lemma A.2 and the space required is $\text{poly}(d)$ by the same lemma. So on the first call to Algorithm 2 a matrix is returned with rows

whose ℓ_p local leverage satisfies $w_i/\text{poly}(d) \leq \hat{w}_i$ (where w_i is the global leverage score and $\hat{w}_i$ is the associated local leverage score) and only those exceeding $\tau/\text{poly}(d)$ are kept.

The algorithm proceeds by repeating this process on a new set of rows from A and an improved matrix B which contains high leverage rows from A already found. Proceeding inductively, we see that when Algorithm 2 is called with matrix $[A'; B]$ then a well-conditioned basis U is computed. Again $[A'; B]_i$ is kept if and only if the local leverage score from U, $w_i(U) > \tau$. By Lemma A.2 this requires $\text{poly}(d)$ space and the local leverage score is at least a $1/\text{poly}(d)$ factor as large as the global leverage score by Lemma A.1. Repeating over all blocks B in A, only the rows of high leverage are kept. Any row of leverage smaller than $\tau/\text{poly}(d)$ is ignored so this is the additive error incurred. □

B. Proofs for Section 4

Algorithm and Discussion

The pseudocode for the first level of the tree structure of the deterministic ℓ_p subspace embedding described in Section 4 is given in Algorithm 3. We use the following notation: m is a counter to index the block of input currently held, denoted $A_{[m]}$, and ranges from 1 to $n^{1-\gamma}$ for the first level of the tree. Similarly, t indexes the current summary, $P^{(t)}$ which are all initialized to be an empty matrix. Again we use the notation $[X; Y]$ to denote the row-wise concatenation of two matrices X and Y with equal column dimension.

Note that Algorithm 3 can be easily distributed as any block of sublinear size can be given to a compute node and then a small-space summary of that block is returned to continue the computation. In addition, the algorithm can be performed using sublinear space in the streaming model because at any one time a summary T of the input can be computed which is of size $d \times d$. Upon reading $A_{[1]}$, a small space summary $P^{(1)}$ is computed and stored with the algorithm proceeding to read in $A_{[2]}$. Similarly, the summary $P^{(2)}$ is computed and if $[P^{(1)}; P^{(1)}]$ does not exceed the storage bound, then the two summaries are merged and this process is repeated until at some point the storage bound is met. Once the summary is large enough that it meets the storage bound, it is then *reduced* by performing the well-conditioned basis reduction (line (5)) and the reduced summary is stored with the algorithm continuing to read and summarize input until a corresponding block in the tree is obtained (or the blocks can be combined to terminate the algorithm).

Proof of Theorem 4.2. Let $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{n^\gamma \times d}$. We compute an ℓ_p well-conditioned basis for B in time $\text{poly}(n^\gamma d)$ by Theorem 2.2; so let $B = US$ for $U \in \mathbb{R}^{n^\gamma \times d}$ and $S \in \mathbb{R}^{d \times d}$ a change of basis matrix.

Algorithm 3 Deterministic ℓ_p subspace embedding

1: **procedure** ℓ_p-SUBSPACEEMBEDDING($A, p, \gamma < 1$))
2: Counters $m, t \leftarrow 1$
3: Summaries $P^{(t)} \leftarrow$ EMPTY for all t.
4: **for** $m = 1 : n^{1-\gamma}$ **do**
5: $A_{[m]} = US$ # U an ℓ_p wcb for A
6: **if** num. rows$(P^{(t)}) + d \leq n^\gamma$ **then**
7: $P^{(t)} \leftarrow [P^{(t)}; S]$
8: **else**
9: $P^{(t+1)} \leftarrow S$
10: $t \leftarrow t + 1$
11: *Merge* all $P^{(t)}$: $T = [P^{(1)}; \ldots; P^{(\cdot)}]$
12: *Reduce* T by splitting into blocks of n^γ and repeating lines (2) - (10) with T in place of A.
13: **return** T

From (Dasgupta et al., 2008), U satisfies $\|x\|_p \leq \|Ux\|_p \leq d\|x\|_p$. This is because $\|x\|_2 \leq \|Ux\|_p \leq \sqrt{d}\|x\|_2$. There are then two cases: if $p < 2$ then

$$\frac{\|x\|_p}{\sqrt{d}} \leq \|x\|_2 \leq \|Ux\|_p \leq \sqrt{d}\|x\|_2 \leq \sqrt{d}\|x\|_p$$

so that $\|x\|_p \leq \|Ux\|_p \leq d\|x\|_p$ by rescaling by $\sqrt{d}$. The third inequality is from (Dasgupta et al., 2008). Similarly, if $p > 2$ then

$$\|x\|_p \leq \|x\|_2 \leq \|Ux\|_p \leq \sqrt{d}\|x\|_2 \leq d\|x\|_p$$

from which $\|x\|_p \leq \|Ux\|_p \leq d\|x\|_p$. Next, the algorithm ignores U and retains only S after computing the well-conditioned basis. Using the above two bounds we readily see that $\|Sx\|_p \leq \|USx\|_p = \|Bx\|_p$. Also, $\|Sx\|_p \geq \|USx\|_p/d = \|Bx\|_p/d$. Now we have obtained a matrix S which satisfies:

$$\frac{\|Bx\|_p}{d} \leq \|Sx\|_p \leq \|Bx\|_p. \tag{14}$$

So $\|Sx\|_p$ agrees with $\|Bx\|_p$ up to a distortion factor of d.

Algorithm 3 applies the *merge and reduce* framework. The matrix A is seen a row at a time and n^γ rows are stored which are used to construct a tree. So at every level a subspace embedding with distortion d is constructed. This error propagates through each of the $O(1/\gamma)$ levels in the tree so the overall distortion to construct the subspace embedding for A is $d^{O(1/\gamma)}$. The space bound is similar; we need $n^\gamma d$ storage per group so require $O(1/\gamma)n^\gamma d$ overall. □

Proof of Theorem 4.3. The task is to minimise $\|Ax - b\|_p$.

Let $Z = [A, b] \in \mathbb{R}^{n\times(d+1)}$ and compute a subspace embedding S for Z using Theorem 4.2. Note that R has $O(1/\gamma)n^\gamma(d+1)$ rows. Let $\Delta = (d+1)^{O(1/\gamma)}$, then for all $y \in \mathbb{R}^{d+1}$ we have:

$$\frac{\|Zy\|_p}{\Delta} \leq \|Sy\| \leq \|Zy\|. \tag{15}$$

Since this condition holds for all $y \in \mathbb{R}^{d+1}$ it must hold, in particular, for vectors $y' = (x, -1)^T$ where $x \in \mathbb{R}^d$ is arbitrary. However, observe that:

$$\|Zy'\|_p = \left\|[A, b]\begin{bmatrix} x \\ -1 \end{bmatrix}\right\|_p = \|Ax - b\|_p. \tag{16}$$

Denote the first d columns of S by $S_{1:d}$ and the last column by S_{d+1}. Then

$$\|Sy'\|_p = \left\|[S_{1:d}, S_{d+1}]\begin{bmatrix} x \\ -1 \end{bmatrix}\right\|_p = \|S_{1:d}x - S_{d+1}\|_p. \tag{17}$$

Now we have transformed the subspace embedding relationship into an instance of regression. In particular, $S_{1:d}$ has only $O(1/\gamma)n^\gamma d$ rows so is a smaller instance than the original problem. We now focus on the task of finding $\min_{x\in\mathbb{R}^d} \|S_{1:d}x - S_{d+1}\|_p$. By using Equation (15) with y' and utilising Equations (16), (17) we have:

$$\frac{\|Ax - b\|_p}{\Delta} \leq \|S_{1:d}x - S_{d+1}\|_p \leq \|Ax - b\|_p. \tag{18}$$

Convex optimisation can now be used to find $\min_{x\in\mathbb{R}^d} \|S_{1:d}x - S_{d+1}\|_p$. Let $\hat{x} = \operatorname{argmin}_{x\in\mathbb{R}^d} \|S_{1:d}x - S_{d+1}\|_p$ which is output from the optimisation and let $x^* = \operatorname{argmin}_{x\in\mathbb{R}^d} \|Ax - b\|_p$ be the optimal solution we would like to estimate. By optimality of $\hat{x}$ we have:

$$\|S_{1:d}\hat{x} - S_{d+1}\|_p \leq \|S_{1:d}x^* - S_{d+1}\|_p. \tag{19}$$

However, combining Equation (19) with Equation (18) we see that:

$$\begin{aligned} \frac{\|A\hat{x} - b\|_p}{\Delta} &\leq \|S_{1:d}\hat{x} - S_{d+1}\|_p && (20)\\ &\leq \|S_{1:d}x^* - S_{d+1}\|_p && (21)\\ &\leq \|Ax^* - b\|_p && (22) \end{aligned}$$

Therefore, $\|A\hat{x} - b\|_p \leq \Delta\|Ax^* - b\|_p$ and $\Delta = \text{poly}(d+1)$ so the ℓ_p-regression problem has been solved up to a polynomial $d+1$ approximation factor. The overall time complexity is the time taken to compute the subspace embedding, which is poly(nd) by Theorem 4.2, and the time for the convex optimisation. However, the optimisation costs poly$(O(1/\gamma)n^\gamma)$ (Woodruff & Zhang, 2013) which is subsumed by the dominant time cost for computing the embedding. Finally, the space cost is immediate from computing the subspace embedding in Theorem 4.2. □

Algorithm 4 Deterministic ℓ_1 low rank approximation (derandomized version of algorithm from (Song et al., 2017))

procedure L1-KRANKAPPROX(X, n, d, k)
  $r = O(k \log k)$
  $m = O(r \log r)$
  $t_1 = O(r \log r)$
  $t_2 = O(m \log m)$
  Generate all diagonal $R \in \mathbb{R}^{d\times d}$ with only r 1s
  Compute all possible sampling and rescaling matrices $D, T_1 \in \mathbb{R}^{n\times n}$ corresponding to Lewis Weights of AR whose entries are powers of 2 between 1 and $1/nd$. There are m and t_1 nonzero entries on the diagonal, respectively.
  Compute all sampling and rescaling matrices $T_2^T \in \mathbb{R}^{d\times d}$ according to the Lewis weights of $(DA)^T$ with t_2 nonzero entries, powers of 2 between 1 and $1/nd$ on the diagonal.
  Evaluate $\|T_1ARXYDAT_2 - T_1AT_2\|_1$ for all choices of above matrices.
  Take the minimal solution
  return ARX, YDA

C. Proofs for Section 5

To prove correctness of Algorithm 5 for Theorem 5.1 we will need to invoke the following algorithm at each level of the tree. This is a derandomized version of an algorithm which returns a low rank approximation to an input matrix. The derandomization follows from generating and testing all possible combinations of the necessary matrices.

Lemma C.1. Algorithm 4 runs in time poly(nd).

Proof. Every matrix which is generated in Algorithm 4 has a number of nonzero entries bounded by $O(k\text{polylog}(k))$. We can test all of the matrices which will take time proportional to the dimension of the matrix (n or d) with exponent $O(k\text{polylog}(k))$ resulting in time poly(nd) overall, since k is constant. □

We need one further lemma which describes the approximation error induced by using well-conditioned bases to decompose a matrix.

Lemma C.2. Let $M \in \mathbb{R}^{N\times D}$ have rank ρ and suppose $U \in \mathbb{R}^{N\times\rho}$ is a well-conditioned basis for M. Let $M = US$ for a change of basis $S \in \mathbb{R}^{\rho\times D}$. Then for all $x \in \mathbb{R}^D$:

$$\frac{\|Sx\|_1}{\text{poly}(D)} \leq \|Mx\|_1 \leq \text{poly}(D)\|Sx\|_1.$$

Proof. For the left-hand side we can just calculate:

$$\begin{aligned}\|Sx\|_1 &\leq D\cdot\|Sx\|_\infty &(23)\\ &\leq D\cdot\text{poly}(D)\|USx\|_1 &(24)\\ &= \text{poly}(D)\cdot\|Mx\|_1. &(25)\end{aligned}$$

The second inequality follows from a property of the well-conditioned basis U. The result follows from observing:

$$\begin{aligned}\|Mx\|_1 = \|USx\|_1 &\leq \|U\|_1\|Sx\|_\infty &(26)\\ &= \text{poly}(D)\|Sx\|_1 &(27)\end{aligned}$$

□

C.1. Proof of Theorem 5.1

For the proof of Theorem 5.1 we introduce Algorithm 5. It is enough to show that for every level, the low rank approximations of each group is polynomially bounded by k in error. The result follows by reasoning how this error grows as we progress through the tree. Denote the jth block of A by $A_{[j]}$.

Algorithm 5 Deterministic ℓ_1 low rank approx

procedure ℓ_1-k-RANKAPPROX(A, k, γ)
  $m, t \leftarrow 1, P_t \leftarrow 0$
  for $i = 1 : 1/\gamma$ **do**
    while $m < n^{1-i\gamma}$ **do**
      while number of rows of $P_m < n^\gamma$ **do**
        Run Algorithm 4 on $A_{[m]}$ and k which outputs matrix $B \in \mathbb{R}^{k\times d}$
        $B \leftarrow WV^T$ (k-rank decomposition)
        Set $W = US$ for well-conditioned basis U
        $P_t \leftarrow SV^T$
        $m \leftarrow m + 1$
  Merge-and-Reduce all P_m until we have an $n^\gamma \times d$ matrix.
  Set P to be matrix of final k rows.
  Solve $\min_Q \|QP - A\|_1$.
  return QP

For every level in the tree we can take a group of rows, C, and perform Algorithm 4. For every C used as input to Algorithm 4 a k-rank matrix B of dimensions $n^\gamma \times d$ is returned. In particular, B has the following property:

$$\|C - B\|_1 \leq \text{poly}(k)\min_{B'\text{rank}k}\|C - B'\|_1. \quad (28)$$

Now factor B using a k rank decomposition. That is, set $B = WV^T$ where W has k columns and V^T has k rows. Further decompose W as $W = US$ for a well-conditioned basis U. Note that W is $n^\gamma \times k$ (and of rank k) by the rank decomposition so U is also $n^\gamma \times k$ and S is $k \times k$. The dimensions of these matrices ensure that individually they do not exceed the space budget from the theorem.

Apply Lemma C.2 with W and k. Then we have for every $x \in \mathbb{R}^k$ that $\|Sx\|_1 = \text{poly}(k)\|Wx\|_1$. Since U is n^γ by k and $k < \text{poly}(d)$, U remains within the required space bound when we use it for the calculation. Now ignore U and store SV^T. Note that each SV^T is a matrix of k directions in $\mathbb{R}^d$. Pass SV^T to the next level of the tree.

Merge the SV^T for each group until we have a matrix of n^γ rows. Repeat the process over all $O(1/\gamma)$ levels in the tree. We require $n^\gamma d$ storage for every group so as we merge and pass SV^T down the levels this combines to total storage of $O(1/\gamma)n^\gamma\text{poly}(d)$. This part of the algorithm is a repeated use of Algorithm 4 which is poly(nd) by Lemma C.1 and some further lower time cost manipulations. Repeating these steps gives poly(nd) as the overall time complexity.

When this is done over all levels we will again have k directions in $\mathbb{R}^d$. Let P be the matrix with these directions as rows. Then we claim that P can be used to construct our approximate ℓ_1 low-rank approximation.

Proposition C.3. Let P be as described above. Then there exists QP which is an ℓ_1 low-rank approximation for A:

$$\min_Q \|QP - A\|_1 \leq \text{poly}(k)\|A - A'\|_1$$

Proof. Each use of Algorithm 4 admits a poly(k) approximation at every level of the tree. Every time the well-conditioned basis U is constructed and then ignored we admit a further poly(k) error due to property 1 of Definition 2.1. The distortion is blown up by a factor of poly(k) every time we use Lemma C.2 which is at every level in the tree. Hence, the total contribution of using Algorithm 4 is $\text{poly}(k)^{O(1/\gamma)} = \text{poly}(k)$ for constant γ. □

Proposition C.3 proves the approximation is poly(k) as claimed. By Lemma C.1 we know that Algorithm 4 is poly(nd) time. The most costly steps in Algorithm 5 are invocations of Algorithm 4 so combining this we see that the overall time cost is poly(nd) as claimed, proving the theorem.

D. Proofs for Section 6

Proof of Theorem 6.1. Given A, the first step is to store all rows of A whose ℓ_p leverage score is above the threshold $\varepsilon/\text{poly}(d)$. This step requires a polynomial increase to $\text{poly}(d)/\varepsilon$ storage from Lemma A.2. Next the change of basis matrix R is computed so that AR is well-conditioned. The stored matrix is B with rows corresponding to those of large ℓ_p leverage scores from A' and zero elsewhere. Also, store all entries in b whose magnitude is greater than $\varepsilon\|b\|_p$ and zero the rest out. Call this vector b'.

We now focus on the task of solving $\min_{x\in\mathbb{R}^d}\|A'Rx - b'\|_\infty$. Any solution must necessarily have $\|x\|_p \leq \text{poly}(d)\|b\|_p$ as otherwise $x = \mathbf{0}$ is a better solution. Recall that $\alpha = d^{1/p+1/2}$ for a well-conditioned basis AR with $p > 2$. Hence, the sum of all the ℓ_p leverage scores is $\alpha^p = d^{O(p)}$. Then the number of rows with leverage score greater than the $\varepsilon/\text{poly}(d)$ is at most $\text{poly}(d)/\varepsilon \cdot d^{O(p)} = \text{poly}(d)/\varepsilon$ for a constant p.

Now, take any row for which the ℓ_p leverage score is less than the $\varepsilon/\text{poly}(d)$ threshold. Then:

$$\begin{aligned}|\langle (AR)_i, x\rangle| &\leq \|(AR)_i\|_\infty \|x\|_1 \\ &\leq \|(AR)_i\|_p \|x\|_1 \\ &\leq \|(AR)_i\|_p \cdot d\|x\|_p \\ &\leq d\frac{\varepsilon}{\text{poly}(d)}\text{poly}(d)\|b\|_p.\end{aligned}$$

By an appropriate choice of the $\text{poly}(d)$ factors scaling ε we see that $|\langle (AR)_i, x\rangle| \leq \varepsilon\|b\|_p$. On such coordinates the ℓ_∞ cost is $|b_i| \pm \varepsilon\|b\|_p$ so by replacing the row with one which is all zero we still pay $|b_i|$ which is within the $\varepsilon\|b\|_p$ had we included the row. The remaining high-leverage score rows are stored in their entirety so the cost on these rows is the same as in the original regression problem. □

Proof of Theorem 6.2. Let S be a set of $2^{\Omega(d)}$ strings in $\{0,1\}^d$ with each coordinate in a string uniformly sampled randomly from $\{0,1\}$. Let $x, y \in S$ and fix a constant $0 < c < 1$. By a Chernoff bound it follows that there are at least cd coordinates in $[d]$ for which $x_i = 0$ and $y_i = 1$ with probability $1 - 2^{-\Omega(d)}$. This implies for appropriate constants in the $\Omega(\cdot)$, by a union bound, all pairs of strings $x, y \in S$ have this property. Hence, such an S exists and we will fix this for the proof.

The regression problem can be reduced to an instance of the `Indexing` problem (Kremer et al., 1999) in data streams as follows. In the stream, the vector b will be all 1s. We will see a random subset T of some elements from S. We claim that it is possible to decide which case we are in: given a random string y, whether y is in S independent of T, or y is in T. This corresponds to solving `Indexing` which requires space $\Omega(|S|) = \Omega(\min\{n, 2^{\Omega(d)}\})$ even with randomization, via communication complexity arguments (Kushilevitz & Nisan, 1997).

Given a test vector y, negate its coordinates so that $y \in \{0,-1\}^d$. Now, append y as a row to the final b coordinate of 1 at the end of the stream to obtain the last item in the stream $(y, 1)$. If y were in S then both y and its complement would be seen as rows of the matrix A. Hence, the optimal cost for ℓ_∞-regression is at least 1. Otherwise, y is not in S. Consider the set of coordinates R where $y_i = 0$. Set $x_i = 1/d$ for $i \in R$ and $-c/2d$ otherwise.

Now we consider the cost of using x. On the row corresponding to the negated vector y the value will be at least $(-1)(-c/2d)(cd) = c^2/2$. Since $b_i = 1$ the cost will be at most $|1 - c^2/2|$ for this coordinate. On all other rows, by using the fact there are at least cd occurrences of $x_i = 0, y_i = 1$ the value will be at least

$$cd(1/d) - (d - cd)(c/2d) \geq c - c/2 = c/2.$$

Hence the cost on these coordinates is at most $|1 - c/2|$. Since $c < 1$, the ℓ_∞ cost is at most $|1 - c^2/2|$. This is a constant factor less than the ℓ_∞ cost of 1 from the previous case so it is possible to decide which of the two cases we are in and hence the space is $\Omega(\min\{n, 2^{\Omega(d)}\})$ as claimed. □

E. Deterministic Approximate Matrix Multiplication

Despite the generality of the subspace embedding result in Theorem 4.2, there may be occasions where the overheads are sufficiently large that it does not make sense to employ this method. One such example is for the *matrix multiplication* problem. Let $A, B \in \mathbb{R}^{n\times d}$ and consider the task of finding a matrix C for which $\|A^TB - C\|_1 < \varepsilon\|A\|_1\|B\|_1$ where $0 < \varepsilon < 1$ and the norm is *entrywise* 1*-norm*.

Lemma E.1. Let $x, y \in \mathbb{R}^n$ have unit entrywise 1-norm. Let $\varepsilon > 0$. Define:

$$\bar{x}_i = \begin{cases} x_i & \text{if } |x_i| > \varepsilon/2, \\ 0 & \text{otherwise,}\end{cases} \qquad \bar{y}_i = \begin{cases} y_i & \text{if } |y_i| > \varepsilon/2, \\ 0 & \text{otherwise.}\end{cases}$$

Then $\langle x, y\rangle - \varepsilon \leq \langle \bar{x}, \bar{y}\rangle \leq \langle x, y\rangle$ and this can be computed using space $O(1/\varepsilon)$.

Proof. Observe that $\bar{x}_i \leq x_i$ and $\bar{y}_i \leq y_i$ for $1 \leq i \leq n$. Hence, $\langle \bar{x}, \bar{y}\rangle \leq \langle x, y\rangle$. For the left-hand side define the following sets: $H_u = \{i : u_i > \varepsilon/2\}, L_u = \{i : u_i \leq \varepsilon/2\}$

for $u = x, y$. Then we can write

$$\begin{aligned}\langle x, y\rangle &= \sum_{i\in H_x} x_i y_i + \sum_{i\in L_x} x_i y_i \\ &= \sum_{i\in H_x\cap H_y} x_i y_i + \sum_{i\in H_x\cap L_y} x_i y_i + \sum_{i\in L_x\cap H_y} x_i y_i + \sum_{i\in L_x\cap L_y} x_i y_i \\ &\leq \langle \bar{x}, \bar{y}\rangle + \frac{\varepsilon}{2}\sum_{i\in H_x} x_i + \frac{\varepsilon}{2}\sum_{i\in H_y} y_i + \frac{\varepsilon}{2}\sum_{i\in L_y} y_i \\ &\leq \langle \bar{x}, \bar{y}\rangle + \varepsilon.\end{aligned}$$

Note that the sum can be written this way as the pair H_x, L_x are disjoint, and likewise for H_y, L_y. The first inequality follows from the second line because $i \in H_x \cap H_y$ means x_i and y_i are retained in $\bar{x}, \bar{y}$ so this summation corresponds directly to $\langle \bar{x}, \bar{y}\rangle$. Then for every $i \in H_x \cap L_y$ we must have that $y_i \leq \varepsilon/2$ so is bounded by $\frac{\varepsilon}{2}\sum_{i\in H_x} x_i$. The same argument holds for the remaining two summations in the inequality. Finally, each of the three summations are at most 1 since both x and y have unit 1-norm. The summations over H_y and L_y when combined are at most the norm of y so can be combined such that $\sum_i y_i \leq 1$. This is enough to prove the result. □

The result for unit vectors is sufficient because we can simply normalize a vector, use Lemma E.1 and then rescale by the norm of x and y. This results in $\langle x, y\rangle - \varepsilon\|x\|_1\|y\|_1 \leq \langle \bar{x}, \bar{y}\rangle \leq \langle x, y\rangle$. This result can be used to prove the following theorem.

Theorem E.2. Let $A, B \in \mathbb{R}^{n\times d}$ and let $\varepsilon > 0$. Let A_i denote the ith row of A and B^i denote the ith column of B. For $X = A$ and $X = B$ define:

$$\overline{X}_{ij} = \begin{cases} X_{ij} & \text{if } |X_{ij}| > \frac{\varepsilon}{2}\|X_i\|_1, \\ 0 & \text{otherwise.}\end{cases}$$

Then in entrywise 1-norm:

$$\|AB^T - \overline{A}\overline{B}^T\|_1 \leq \varepsilon\|A\|_1\|B\|_1.$$

Proof. Fix $\varepsilon > 0$. The matrix product takes a row of A with a column of B^T which is simply a row of B. These are both vectors in $\mathbb{R}^d$ so we can apply the transformation as in the Theorem statement, which is equivalent to that in Lemma E.1. By applying the rescaled version of Lemma E.1 we see that:

$$|\langle A_i, B_j\rangle - \langle \overline{A}_i, \overline{B}_j\rangle \leq \varepsilon\|A_i\|_1\|B_j\|_1. \tag{29}$$

Now the norm $\|AB^T - \overline{A}\overline{B}^T\|_1$ is the sum of all summands defined as in Equation 29 over all pairs of i and j. Computing the sum then gives the desired result. □

The argument from Lemma E.1 can easily be adapted to obtain a result for the matrix profuct A^TB. Observe that approximating A^TB is equivalent to approximating inner products between columns of A and columns of B. The modification is that the summary must be applied column-wise instead of row-wise as in Theorem E.2.

Theorem E.3. Let $A, B \in \mathbb{R}^{n\times d}$ and let $\varepsilon > 0$. Then there exists a deterministic algorithm which uses $O(1/\varepsilon)$ space and outputs $\overline{A}$ and $\overline{B}$ which satisfy:

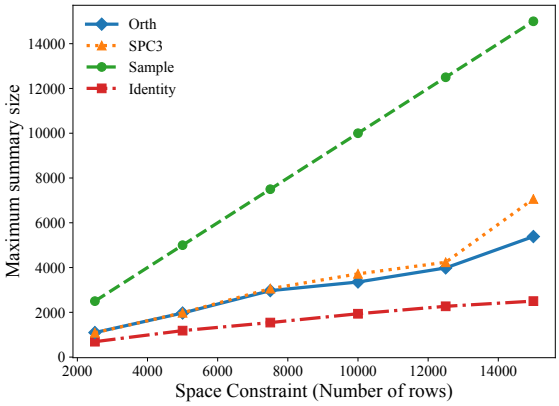
$$\|A^TB - \overline{A}^T\overline{B}\|_1 \leq \varepsilon\|A\|_1\|B\|_1.$$

Proof. For a matrix X let X_i denote the ith row and X^j denote the jth column. Let $\|X^j_{:i}\|_1$ denote the 1-norm of column j of X up to and including row i. It is clear that this norm is monotonic as more rows are seen in the stream. In particular, $\|X^j_{:n}\|_1 = \|X\|_1$. Therefore, the algorithm can be modified as follows: upon seeing a row i, if $|X_{ij}| > \varepsilon/2 \cdot \|X^j_{:i}\|_1$ then keep the entry X_{ij} and otherwise set $X_{oj} = 0$. It is sufficient to consider only the last row. At this stage all rows which have not exceeded the running threshold upon seeing a particular row will have been ignored and only those which exceed $\varepsilon/2 \cdot \|X^j_{:n-1}\|_1$ will be stored. Then by increasing the threshold upon seeing row n only the X_{ij} which exceed $\|X^j_{:n}\|_1 = \|X\|_1$ will be kept and this is exactly the same set of rows as had the summary been applied given full access to the rows.

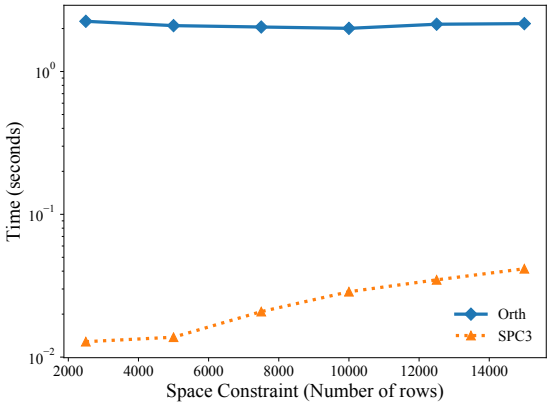
Hence, we may apply the result from Lemma E.1 on the columns of A and B as described above. It is then straightforward to show in a similar way to the lemma that the claim of the theorem holds. □

F. Further Experimental results

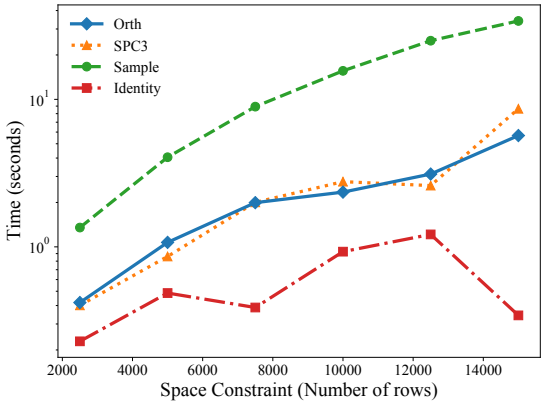
Here we illustrate the remaining experimental results on the YearPredictionMSD dataset which include the space and time plots. The experimental setup is the same as outline in Section 7.



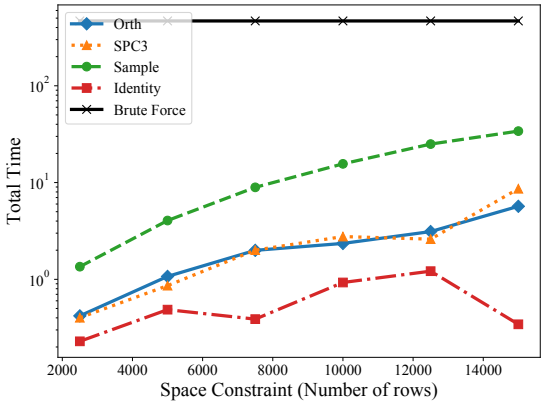
(a) Summary Size



(b) Update Time



(c) Query Time



(d) Total Time

Figure 4: Remaining plots for YearPredictionMSD data.