Coordinate Methods for Matrix Games

Yair Carmon*†, Yujia Jin†, Aaron Sidford†, Kevin Tian†
*Tel Aviv University †Stanford University
{yairc, yujiajin, sidford, kjtian}@stanford.edu

***Abstract*—We develop primal-dual coordinate methods for solving bilinear saddle-point problems of the form $\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} y^\top Ax$ which contain linear programming, classification, and regression as special cases. Our methods push existing fully stochastic sublinear methods and variance-reduced methods towards their limits in terms of per-iteration complexity and sample complexity. We obtain nearly-constant per-iteration complexity by designing efficient data structures leveraging Taylor approximations to the exponential and a binomial heap. We improve sample complexity via low-variance gradient estimators using dynamic sampling distributions that depend on both the iterates and the magnitude of the matrix entries.**

Our runtime bounds improve upon those of existing primal-dual methods by a factor depending on *sparsity measures* of the m by n matrix A. For example, when rows and columns have constant ℓ_1/ℓ_2 norm ratios, we offer improvements by a factor of $m+n$ in the fully stochastic setting and $\sqrt{m+n}$ in the variance-reduced setting. We apply our methods to computational geometry problems, i.e. minimum enclosing ball, maximum inscribed ball, and linear regression, and obtain improved complexity bounds. For linear regression with an elementwise nonnegative matrix, our guarantees improve on exact gradient methods by a factor of $\sqrt{\mathsf{nnz}(A)/(m+n)}$.

***Keywords*-minimax optimization; stochastic gradient methods; matrix games; linear regression**

This version is a preliminary extended abstract, and a full version [1] can be found at https://arxiv.org/abs/2009.08447.

I. Introduction

Bilinear minimax problems of the form

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} y^\top Ax \text{ where } A\in\mathbb{R}^{m\times n}, \tag{1}$$

are fundamental to machine learning, economics and theoretical computer science [2–4]. We focus on three important settings characterized by different domain geometries. When $\mathcal{X}$ and $\mathcal{Y}$ are probability simplices—which we call the ℓ_1-ℓ_1 setting—the problem (1) corresponds to a zero-sum matrix game and also to a linear program in canonical feasibility form. The ℓ_2-ℓ_1 setting, where $\mathcal{X}$ is a Euclidean ball and $\mathcal{Y}$ is a simplex, is useful for linear classification (hard-margin support vector machines) as well as problems in computational geometry [5]. Further, the ℓ_2-ℓ_2 setting, where both $\mathcal{X}$ and $\mathcal{Y}$ are Euclidean balls (with general center), includes linear regression.

Many problems of practical interest are *sparse*, i.e., the number of nonzero elements in A, which we denote by nnz, satisfies $\mathsf{nnz} \ll mn$. Examples include: linear programs with constraints involving few variables, linear classification with 1-hot-encoded features, and linear systems that arise from physical models with local interactions. The problem description size nnz plays a central role in several runtime analyses of algorithms for solving the problem (1).

However, sparsity is not an entirely satisfactory measure of instance complexity: it is not continuous in the elements of A and consequently it cannot accurately reflect the simplicity of "nearly sparse" instances with many small (but nonzero) elements. Measures of *numerical sparsity*, such as the ℓ_1 to ℓ_2 norm ratio, can fill this gap [6]. Indeed, many problems encountered in practice are numerically sparse. Examples include: linear programming constraints of the form $x_1 \geq \frac{1}{n}\sum_i x_i$, linear classification with neural network activations as features, and linear systems arising from physical models with interaction whose strength decays with distance.

Existing bilinear minimax solvers do not exploit the numerical sparsity of A and their runtime guarantees do not depend on it—the basic limitation of these methods is that they do not directly access the large matrix entries, and instead sample the full columns and rows in which they occur. To overcome this limitation, we propose methods that access A a single entry at a time, leverage numerical sparsity by accessing larger coordinates more frequently, and enjoy runtime guarantees that depend explicitly on numerical sparsity measures. For numerically sparse large-scale instances our runtimes are substantially better than the previous state-of-the-art. Moreover, our runtimes subsume the previous state-of-the-art dependence on nnz and rcs, the maximum number of nonzeros in any row or column.

In addition to proposing algorithms with improved runtimes, we develop two techniques that may be of broader interest. First, we design non-uniform sampling schemes that minimize regret bounds; we use a general framework that unifies the Euclidean and (local norms) simplex geometries, possibly facilitating future extension. Second, we build a data structure capable of efficiently maintaining and sampling from multiplicative weights iterations (i.e. entropic projection) *with a fixed dense component*. This data structure overcomes limitations of existing techniques for maintaining entropic projections and we believe it may prove effective in other settings where such projections appear.

Table I
DEPENDENCE ON A FOR DIFFERENT METHODS IN DIFFERENT GEOMETRIES.

	L_{mv} (matrix-vector)	L_{rc} (row-column)	L_{co} (coordinate)
ℓ_1-ℓ_1	$\max_{i,j} \lvert A_{ij}\rvert$	$\max_{i,j} \lvert A_{ij}\rvert$	$\max\left\{ \max_i \left\lVert A_{i:}\right\rVert_2, \max_j \left\lVert A_{:j}\right\rVert_2 \right\}$
ℓ_2-ℓ_1	$\max_i \left\lVert A_{i:}\right\rVert_2$	$\max_i \left\lVert A_{i:}\right\rVert_2$	$\max\left\{ \max_i \left\lVert A_{i:}\right\rVert_1, \lVert A\rVert_{\mathrm{F}} \right\}^{\dagger}$
ℓ_2-ℓ_2	$\lVert A\rVert_{\mathrm{op}}$	$\lVert A\rVert_{\mathrm{F}}$	$\max\left\{ \sqrt{\sum_i \lVert A_{i:}\rVert_1^2}, \sqrt{\sum_j \lVert A_{:j}\rVert_1^2} \right\}$

Comments: $A_{i:}$ and $A_{:j}$ are the ith row and jth column of A, respectively. Numerically sparse instances satisfy $L_{\mathsf{co}} = O(L_{\mathsf{rc}})$. † In the ℓ_2-ℓ_1 setting we can also achieve, via alternative sampling schemes, $L_{\mathsf{co}} = L_{\mathsf{rc}}\sqrt{\mathsf{rcs}}$ and $L_{\mathsf{co}} = \max\{\max_i \lVert A_{i:}\rVert_1, \sqrt{\max_i \lVert A_{i:}\rVert_1 \max_j \lVert A_{:j}\rVert_1}\}$.

Table II
COMPARISON OF ITERATIVE METHODS FOR BILINEAR PROBLEMS

Method	Iteration cost	Total runtime
Exact gradient [7, 8]	$O(\mathsf{nnz})$	$\widetilde{O}\left(\mathsf{nnz} \cdot L_{\mathsf{mv}} \cdot \epsilon^{-1}\right)$
Row-column [5, 9, 10]	$O(n+m)$	$\widetilde{O}\left((m+n) \cdot L_{\mathsf{rc}}^2 \cdot \epsilon^{-2}\right)$
Row-column VR [10, 11]	$O(n+m)$	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz} \cdot (m+n)} \cdot L_{\mathsf{rc}} \cdot \epsilon^{-1}\right)$
Sparse row-col (folklore)	$\widetilde{O}(\mathsf{rcs})$	$\widetilde{O}\left(\mathsf{rcs} \cdot L_{\mathsf{rc}}^2 \cdot \epsilon^{-2}\right)$
Sparse row-col VR (this work)	$\widetilde{O}(\mathsf{rcs})$	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz} \cdot \mathsf{rcs}} \cdot L_{\mathsf{rc}} \cdot \epsilon^{-1}\right)$
Coordinate (this work)	$\widetilde{O}(1)$	$\widetilde{O}\left(\mathsf{nnz} + L_{\mathsf{co}}^2 \cdot \epsilon^{-2}\right)$
Coordinate VR (this work)	$\widetilde{O}(1)$	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz}} \cdot L_{\mathsf{co}} \cdot \epsilon^{-1}\right)$

Comments: nnz denotes the number of nonzeros in $A \in \mathbb{R}^{m\times n}$ and $\mathsf{rcs} \le \max\{m, n\}$ denotes the maximum number of nonzeros in any row and column of A. The quantities $L_{\mathsf{mv}}, L_{\mathsf{co}}$ and L_{rc} depend on problem geometry (see Table I).

Table III
COMPARISON OF COMPLEXITY FOR DIFFERENT APPLICATIONS.

Task	Method	Runtime $(m \ge n)$
MaxIB	Allen-Zhu et al. [12]	$\widetilde{O}\left(mn + \rho m\sqrt{n} \cdot \epsilon^{-1}\right)$
	Our method (this work)	$\widetilde{O}\left(\mathsf{nnz} + \rho\sqrt{\mathsf{nnz} \cdot \mathsf{rcs}} \cdot \epsilon^{-1}\right)$
MinEB	Allen-Zhu et al. [12]	$\widetilde{O}\left(mn + m\sqrt{n} \cdot \epsilon^{-1/2}\right)$
	Our method (this work)	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz} \cdot \mathsf{rcs}} \cdot \epsilon^{-1/2}\right)^{\dagger}$
Regression $(A^\top A \succeq \mu I)$	AGD [13]	$\widetilde{O}\left(\mathsf{nnz} \cdot \lVert A\rVert_{\mathrm{op}} \frac{1}{\sqrt{\mu}}\right)$
	Gupta and Sidford [6]	$\widetilde{O}\left(\mathsf{nnz} + \mathsf{nnz}^{2/3} \cdot \left(\sum_{i\in[n]} \lVert A\rVert_{\mathrm{F}} \cdot \lVert A_{i:}\rVert_1 \cdot \lVert A_{i:}\rVert_2\right)^{1/3} \frac{1}{\sqrt{\mu}}\right)$
	Our method (this work)	$\widetilde{O}\left(\mathsf{nnz} + \sqrt{\mathsf{nnz}} \cdot \max\left\{\sqrt{\sum_i \lVert A_{i:}\rVert_1^2}, \sqrt{\sum_j \lVert A_{:j}\rVert_1^2}\right\} \frac{1}{\sqrt{\mu}}\right)$

Comments: ρ denotes the radii ratio of the minimum ball enclosing the rows of A and maximum ball inscribed in them. † For MinEB, we state an upper bound here and refer readers to the full version for a more fine-grained runtime.

A. *Our results*

Table II summarizes our runtime guarantees and puts them in the context of the best existing results. We consider methods that output (expected) ϵ-accurate solutions of the saddle-point problem (1), namely a pair x, y satisfying

$$\mathbb{E}\left[\max_{v\in\mathcal{Y}} v^\top Ax - \min_{u\in\mathcal{X}} y^\top Au\right] \le \epsilon.$$

The algorithms in Table II are all iterative solvers for the general problem $\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} f(x,y)$, specialized to $f(x,y) = y^\top Ax$. Each algorithm presents a different tradeoff between per-iteration complexity and the required iteration count, corresponding to the matrix access modality: exact gradient methods compute matrix-vector products in each iteration, row-column stochastic gradient methods sample a row and a column in each iteration, and our proposed coordinate methods take this tradeoff to an extreme by sampling a single coordinate of the matrix per iteration.[1] In addition, variance reduction (VR) schemes combine both fast stochastic gradient computations and infrequent exact gradient computations, maintaining the amortized per-iteration cost of the stochastic scheme and reducing the total iteration count for sufficiently small ϵ.

The runtimes in Table II depend on the numerical range of A through a matrix norm L that changes with both the problem geometry and the type of matrix access; we use L_{mv}, L_{rc} and L_{co} to denote the constants corresponding to matrix-vector products, row-column queries and coordinated queries, respectively. Below, we describe these runtimes in detail. In the settings we study, our results are the first theoretical demonstration of runtime gains arising from sampling a single coordinate of A at a time, as opposed to entire rows and columns.

Coordinate stochastic gradient methods. We develop coordinate stochastic gradient estimators which allow per-iteration cost $\widetilde{O}(1)$ and iteration count $\widetilde{O}\left(n+m+(\frac{L_{\mathsf{co}}}{\epsilon})^2\right)$. We define L_{co} in Table I; for each domain geometry, the quantity $\frac{L_{\mathsf{co}}}{L_{\mathsf{rc}}}$ is a measure of the numerical sparsity of A, satisfying

$$1 \le \frac{L_{\mathsf{co}}^2}{L_{\mathsf{rc}}^2} \le \mathsf{rcs}.$$

Every iteration of our method requires sampling an element in a row or a column with probability proportional to its entries. Assuming a matrix access model that allows such sampling in time $\widetilde{O}(1)$ (similarly to [17–19]), the total runtime of our method is $\widetilde{O}\left(n+m+(\frac{L_{\mathsf{co}}}{\epsilon})^2\right)$. In this case, for numerically sparse problems such that $L_{\mathsf{co}} = O(L_{\mathsf{rc}})$, the proposed coordinate methods outperform row-column sampling by a factor of $m+n$. Moreover, the bound $L_{\mathsf{co}}^2 \le L_{\mathsf{rc}}^2(m+n)$ implies that our runtime is never worse than that of row-column methods. When only coordinate access to the matrix A is initially available, we may implement the required sampling access via preprocessing in time $O(\mathsf{nnz})$. This changes the runtime to $\widetilde{O}\left(\mathsf{nnz}+(\frac{L_{\mathsf{co}}}{\epsilon})^2\right)$, so that the comparison above holds only when $(\frac{L_{\mathsf{co}}}{\epsilon})^2 = \tilde{\Omega}(\mathsf{nnz})$. In that regime, the variance reduction technique we describe below provides even stronger guarantees.

Coordinate methods with variance reduction. Using our recently proposed framework [11] we design a variance reduction algorithm with amortized per-iteration cost $\widetilde{O}(1)$, required iteration count of $\widetilde{O}\left(\sqrt{\mathsf{nnz}}\cdot\frac{L_{\mathsf{co}}}{\epsilon}\right)$ and total running time $\widetilde{O}\left(\mathsf{nnz}+\sqrt{\mathsf{nnz}}\cdot\frac{L_{\mathsf{co}}}{\epsilon}\right)$. In the numerically sparse regime $L_{\mathsf{co}} = O(L_{\mathsf{rc}})$, our runtime improves on row-column VR by a factor of $\sqrt{\mathsf{nnz}/(m+n)}$, and in general the bound $L_{\mathsf{co}} \le L_{\mathsf{rc}}\sqrt{m+n}$ guarantees it is never worse. Since variance reduction methods always require a single pass over the data to compute an exact gradient, this comparison holds regardless of the matrix access model. In the ℓ_2-ℓ_2 setting we note that for *elementwise non-negative* matrices, $L_{\mathsf{co}} = \max\{\|A\mathbf{1}\|_2, \|A^\top\mathbf{1}\|_2\} \le L_{\mathsf{mv}}\sqrt{m+n}$, and consequently our method outperforms exact gradient methods by a factor of $\sqrt{\mathsf{nnz}/(m+n)}$, even without any numerical or spectral sparsity in A. Notably, this is the same factor of improvement that row-column VR achieves over exact gradient methods in the ℓ_1-ℓ_1 and ℓ_2-ℓ_1 regimes.

Optimality of the constant L_{co}. For the ℓ_1-ℓ_1 and ℓ_2-ℓ_2 settings, we argue that the constant L_{co} in Table I is optimal in the restricted sense that no alternative sampling distribution for coordinate gradient estimation can have a better variance bound than L_{co} (a similar sense of optimality also holds for L_{rc} in each geometry). In the ℓ_2-ℓ_1 setting, a different sampling distribution produces an improved (and optimal) constant $\max\{\max_i \|A_{i:}\|_1, \||A|\|_{\mathrm{op}}\}$, where $A_{i:}$ is the ith row of A, and $|A|_{ij} = |A_{ij}|$ is the elementwise absolute value of A. However, it is unclear how to efficiently sample from this distribution.

Row-column sparse instances. Some problem instances admit a structured form of sparsity where every row and column has at most rcs nonzero elements. In all settings we have $L_{\mathsf{co}} \le L_{\mathsf{rc}}\sqrt{\mathsf{rcs}}$ and so our coordinate methods naturally improve when rcs is small. Specifically, the sampling distributions and data structures we develop in this paper allow us to modify previous methods for row-column VR [11] to leverage row-column sparsity, reducing the amortized per-iteration cost from $O(m+n)$ to $\widetilde{O}(\mathsf{rcs})$.

Applications. We illustrate the implications of our results for two problems in computational geometry, mini-

[1] Interior point methods offer an alternative tradeoff between iteration cost and iteration count: the number of required iterations depends on $1/\epsilon$ only logarithmically, but every iteration is costly, requiring a linear system solution which at present takes time $\Omega(\min\{m,n\}^2)$. In the ℓ_1-ℓ_1 geometry, the best known runtimes for interior point methods are $\widetilde{O}((\mathsf{nnz}+\min\{m,n\}^2)\sqrt{\min\{m,n\}})$ [14], $\widetilde{O}(\max\{m,n\}^\omega)$ [15], and $\widetilde{O}(mn+\min\{m,n\}^3)$ [16]. In this paper we are mainly interested in the large-scale low-accuracy regime with $L/\epsilon < \min(m,n)$ where the runtimes described in Table II are favorable (with the exception of [16] in certain cases). Our methods take only few passes over the data, which are not the case for many interior-point methods [14, 15]. Also, our methods do not rely on a general (ill-conditioned) linear system solver, which is a key ingredient in interior point methods.

mum enclosing ball (Min-EB) and maximum inscribed ball (Max-IB), as well as linear regression. For Min-EB and Max-IB in the non-degenerate case $m \geq n$, we apply our ℓ_2-ℓ_1 results to obtain algorithms whose runtime bounds coincide with the state-of-the-art [12] for dense problems, but can be significantly better for sparse or row-column sparse instances. For linear regression we focus on accelerated linearly converging algorithms, i.e., those that find x such that $\|Ax - b\|_2 \leq \epsilon$ in time proportional to $\mu^{-\frac{1}{2}} \log \frac{1}{\epsilon}$ where μ is the smallest eigenvalue of $A^\top A$. Within this class and in a number of settings, our reduced variance coordinate method offers improvement over the state-of-the-art: for instances where $\|A_{i:}\|_1 = O(\|A_{i:}\|_2)$ and $\|A_{:j}\|_1 = O(\|A_{:j}\|_2)$ for all i, j it outperforms [6] by a factor of $\mathsf{nnz}^{1/6}$, and for elementwise nonnegative instances it outperforms accelerated gradient descent by a factor of $\sqrt{\mathsf{nnz}/(m+n)}$. See Table III for a detailed runtime comparison.

B. Our approach

We now provide a detailed overview of our algorithm design and analysis techniques, highlighting our main technical insights. We focus on the ℓ_1-ℓ_1 geometry, since it showcases all of our developments. Our technical contributions have two central themes:

1) **Sampling schemes design.** The key to obtaining efficient coordinate methods is carefully choosing the sampling distribution. Here, local norms analysis of stochastic mirror descent [20] on the one hand enables tight regret bounds, and on the other hand imposes an additional design constraint since the stochastic estimators must be bounded for the analysis to apply. We achieve estimators with improved variance bounds meeting this boundedness constraint by leveraging a "clipping" operation introduced by Clarkson et al. [5]. Specifically, in the simplex geometry, we truncate large coordinates of our estimators, and show that our method is robust to the resulting distortion.
2) **Data structure design.** Our goal is to perform iterations in $\widetilde{O}(1)$ time, but our mirror descent procedures call for updates that change $m + n$ variables in each step. We resolve this tension via data structures that implicitly maintain the iterates. Variance reduction poses a considerable challenge here, because every reduced-variance stochastic gradient contains a dense component that changes all coordinates in a complicated way. In particular, existing data structures cannot efficiently compute the normalization factor necessary for projection to the simplex. We design a data structure that overcomes this hurdle via Taylor expansions, coordinate binning, and a binomial heap-like construction. The data structure computes approximate mirror projections, and we modify the standard mirror descent analysis to show it is stable under the particular structure of the resulting approximation errors.

At the intersection of these two themes is a novel sampling technique we call "sampling from the sum," which addresses the same variance challenges as the "sampling from the difference" technique of [11], but is more amenable to efficient implementation with a data structure.

1) Coordinate stochastic gradient method Our algorithm is an instance of stochastic mirror descent [21], which in the ℓ_1-ℓ_1 setting produces a sequence of iterates $(x_1, y_1), (x_2, y_2), \ldots$ according to

$$\begin{aligned} x_{t+1} &= \Pi_\Delta\left(x_t \circ \exp\{-\eta \tilde{g}^{\mathsf{x}}(x_t, y_t)\}\right) \\ \text{and } \; y_{t+1} &= \Pi_\Delta\left(y_t \circ \exp\{-\eta \tilde{g}^{\mathsf{y}}(x_t, y_t)\}\right), \end{aligned} \tag{2}$$

where $\Pi_\Delta(v) = \frac{v}{\|v\|_1}$ is the projection onto the simplex (exp and log are applied to vectors elementwise, and elementwise multiplication is denoted by $\circ$), η is a step size, and $\tilde{g}^{\mathsf{x}}, \tilde{g}^{\mathsf{y}}$ are stochastic gradient estimators for $f(x,y) = y^\top A x$ satisfying

$$\begin{aligned} \mathbb{E}\, \tilde{g}^{\mathsf{x}}(x,y) &= \nabla_x f(x,y) = A^\top y \\ \text{and } \; \mathbb{E}\, \tilde{g}^{\mathsf{y}}(x,y) &= -\nabla_y f(x,y) = -Ax. \end{aligned}$$

We describe the computation and analysis of $\tilde{g}^{\mathsf{x}}$; the treatment of $\tilde{g}^{\mathsf{y}}$ is analogous. To compute $\tilde{g}^{\mathsf{x}}(x,y)$, we sample i, j from a distribution $p(x,y)$ on $[m] \times [n]$ and let

$$\tilde{g}^{\mathsf{x}}(x,y) = \frac{y_i A_{ij}}{p_{ij}(x,y)} e_j, \tag{3}$$

where $p_{ij}(x,y)$ denotes the probability of drawing i, j from $p(x,y)$ and e_j is the jth standard basis vector—a simple calculation shows that $\mathbb{E}\, \tilde{g}^{\mathsf{x}} = A^\top y$ for any p. We first design $p(x,y)$ to guarantee an $\widetilde{O}\left((\frac{L_{\mathsf{co}}}{\epsilon})^2\right)$ iteration complexity for finding an ϵ-accurate solution, and then briefly touch on how to compute the resulting iterations in $\widetilde{O}(1)$ time.

Local norms-informed distribution design. The standard stochastic mirror descent analysis [21] shows that if $\mathbb{E}\, \|\tilde{g}^{\mathsf{x}}(x,y)\|_\infty^2 \leq L^2$ for all x, y (and similarly for $\tilde{g}^{\mathsf{y}}$), taking $\eta = \frac{\epsilon}{L^2}$ and a choice of $T = \widetilde{O}\left((\frac{L}{\epsilon})^2\right)$ suffices to ensure that the iterate average $\frac{1}{T}\sum_{t=1}^T (x_t, y_t)$ is an ϵ-accurate solution in expectation. Unfortunately, this analysis demonstrably fails to yield sufficiently tight bounds for our coordinate estimator: there exist instances for which any distribution p produces $L \geq n L_{\mathsf{rc}}$. We tighten the analysis using a *local norms* argument [cf. 20, Section 2.8], showing that $\widetilde{O}\left((\frac{L}{\epsilon})^2\right)$ iterations suffice whenever $\|\eta \tilde{g}^{\mathsf{x}}\|_\infty \leq 1$ with probability 1 and for all x, y

$$\mathbb{E}\, \|\tilde{g}^{\mathsf{x}}(x,y)\|_x^2 \leq L^2, \quad \text{where} \quad \|\gamma\|_x^2 = \sum_j x_j \gamma_j^2$$

is the local norm at $x \in \mathcal{X}$. We take

$$p_{ij} = y_i \frac{A_{ij}^2}{\|A_{i:}\|_2^2} \tag{4}$$

(recalling that x, y are both probability vectors). Substituting into (3) gives

$$\begin{aligned}\mathbb{E}\,\|\tilde{g}^{\mathsf{x}}(x,y)\|_x^2 &= \sum_{i,j}\frac{y_i^2 A_{ij}^2 x_j}{p_{ij}} = \sum_{i,j} y_i \|A_{i:}\|_2^2\, x_j \\ &= \sum_i y_i \|A_{i:}\|_2^2 \le \max_i \|A_{i:}\|_2^2 \le L_{\mathsf{co}}^2,\end{aligned}$$

with $L_{\mathsf{co}} = \max\{\max_i \|A_{i:}\|_2, \max_j \|A_{:j}\|_2\}$ as in Table I.

While this is the desired bound on $\mathbb{E}\,\|\tilde{g}^{\mathsf{x}}(x,y)\|_x^2$, the requirement $\|\eta\tilde{g}^{\mathsf{x}}\|_\infty \le 1$ does not hold when A has sufficiently small elements. We address this by clipping $\tilde{g}$: we replace $\eta\tilde{g}^{\mathsf{x}}$ with $\mathrm{clip}(\eta\tilde{x}^{\mathsf{x}})$, where

$$[\mathrm{clip}(v)]_i := \min\{|v_i|, 1\}\,\mathrm{sign}(v_i),$$

the Euclidean projection to the unit box. The clipped gradient estimator clearly satisfies the desired bounds on infinity norm and local norm second moment, but is biased for the true gradient. Following the analysis of Clarkson et al. [5], we account for the bias by relating it to the second moment via

$$|\,\langle \gamma - \mathrm{clip}(\gamma), x\rangle\,| \le \|\gamma\|_x^2,$$

which allows to absorb the effect of the bias into existing terms in our error bounds. Putting together these pieces yields the desired bound on the iteration count.

Efficient implementation. Data structures for performing the update (2) and sampling from the resulting iterates in $\widetilde{O}(1)$ time are standard in the literature [e.g., 22]. We add to these the somewhat non-standard ability to also efficiently track the running sum of the iterates. To efficiently sample $i, j \sim p$ according to (4) we first use the data structure to sample $i \sim y$ in $\widetilde{O}(1)$ time and then draw $j \in [n]$ with probability proportional to A_{ij}^2 in time $O(1)$, via either $O(\mathsf{nnz})$ preprocessing or an appropriate assumption about the matrix access model. The "heavy lifting" of our data structure design is dedicated for supporting variance reduction, which we describe in the next section.

Sampling distributions beyond ℓ_1-ℓ_1. Table IV lists the sampling distributions we develop for the various problem geometries. Note that for the ℓ_2-ℓ_1 setting we give three different distributions for sampling the simplex block of the gradient (i.e., $\tilde{g}^{\mathsf{y}}$); each distribution corresponds to a different parameter L_{co} (see comments following Table I). The distribution $q_{ij} \propto \sqrt{y_i}\,|A_{ij}x_j|$ yields a stronger bound L in the ℓ_2-ℓ_1 setting, but we do not know how to efficiently sample from it.

2) Coordinate variance reduction To accelerate the stochastic coordinate method we apply our recently proposed variance reduction framework [11]. This framework operates in $\frac{\alpha}{\epsilon}$ epochs, where α is a design parameter that trades between full and stochastic gradient computations. Each epoch consists of three parts: (i) computing the exact gradient at a reference point (x_0, y_0), (ii) performing T iterations of regularized stochastic mirror descent to produce the sequence $(x_1, y_1), \ldots, (x_T, y_T)$ and (iii) taking an extra-gradient step from the average of the iterates in (ii). Setting $\kappa = 1/(1 + \eta\alpha/2)$, the iterates x_t follow the recursion

$$\begin{aligned} x_{t+1} &= \Pi_\Delta\big(x_t^\kappa \circ x_0^{1-\kappa} \circ \exp\{-\eta\kappa[g_0^{\mathsf{x}} + \tilde{\delta}^{\mathsf{x}}(x_t, y_t)]\}\big), \\ &\text{where } \Pi_\Delta(v) = \frac{v}{\|v\|_1}, \text{ and } g_0^{\mathsf{x}} = A^\top y_0 \end{aligned} \tag{5}$$

is the exact gradient at the reference point, and $\tilde{\delta}^{\mathsf{x}}$ is a stochastic gradient *difference* estimator satisfying

$$\mathbb{E}\,\tilde{\delta}^{\mathsf{x}}(x,y) = \nabla_x f(x,y) - \nabla_x f(x_0, y_0) = A^\top(y - y_0).$$

The iteration for y_t is similar. In [11] we show that if $\tilde{\delta}^{\mathsf{x}}$ satisfies

$$\mathbb{E}\,\|\tilde{\delta}^{\mathsf{x}}(x,y)\|_\infty^2 \le L^2\left(\|x - x_0\|_1^2 + \|y - y_0\|_1^2\right) \quad \forall x, y \tag{6}$$

and a similar bound holds on $\mathbb{E}\,\|\tilde{\delta}^{\mathsf{y}}(x,y)\|_\infty^2$, then $T = O(\frac{L^2}{\alpha^2})$ iterations per epoch with step size $\eta = \frac{\alpha}{L^2}$ suffice for the overall algorithm to return a point with expected error below ϵ.

We would like to design a coordinate-based estimator $\tilde{\delta}$ such that the bound (6) holds for $L = L_{\mathsf{co}}$ as in Table I and each iteration (5) takes $\widetilde{O}(1)$ time. Since every epoch also requires $O(\mathsf{nnz})$ time for matrix-vector product (exact gradient) computations, the overall runtime would be $\widetilde{O}((\mathsf{nnz} + \frac{L_{\mathsf{co}}^2}{\alpha^2}) \cdot \frac{\alpha}{\epsilon})$. Choosing $\alpha = L_{\mathsf{co}}/\sqrt{\mathsf{nnz}}$ then gives the desired runtime $\widetilde{O}(\mathsf{nnz} + \sqrt{\mathsf{nnz}} \cdot \frac{L_{\mathsf{co}}}{\epsilon})$.

Distribution design (sampling from the difference). We start with a straightforward adaptation of the general estimator form (3). To compute $\tilde{\delta}^{\mathsf{x}}(x,y)$, we sample $i, j \sim p$, where p may depend on x, x_0, y and y_0, and let

$$\tilde{\delta}^{\mathsf{x}}(x,y) = \frac{(y_i - [y_0]_i)A_{ij}}{p_{ij}} e_j, \tag{7}$$

where e_j is the jth standard basis vector. As in the previous section, we find that the requirement (6) is too stringent for coordinate-based estimators. Here too, we address this challenge with a local norms argument and clipping of the difference estimate. Using the "sampling from the difference" technique from [11], we arrive at

$$p_{ij} = \frac{|y_i - [y_0]_i|}{\|y - y_0\|_1} \cdot \frac{A_{ij}^2}{\|A_{i:}\|_2^2}. \tag{8}$$

This distribution satisfies the local norm relaxation of (6) with $L^2 = L_{\mathsf{co}}^2$.

Data structure design. Efficiently computing (5) is significantly more challenging than its counterpart (2). To clarify the difficulty and describe our solution, we write

$$x_t = \Pi_\Delta(\hat{x}_t) = \hat{x}_t / \|\hat{x}_t\|_1$$

Table IV
THE DISTRIBUTIONS USED IN OUR COORDINATE GRADIENT ESTIMATOR

Setting	p_{ij}	q_{ij}
ℓ_1-ℓ_1	$y_i \cdot \frac{A_{ij}^2}{\lVert A_{i:}\rVert_2^2}$	$x_j \cdot \frac{A_{ij}^2}{\lVert A_{:j}\rVert_2^2}$
ℓ_2-ℓ_1	$y_i \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\frac{A_{ij}^2}{\lVert A\rVert_{\mathrm{F}}^2}$
ℓ_2-ℓ_1	$y_i \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\propto x_j^2 \cdot \mathbf{1}_{A_{ij}\neq 0}$
ℓ_2-ℓ_1	$y_i \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\frac{A_{ij} \cdot x_j^2}{\sum_{k\in[n]} \lVert A_{:k}\rVert_1 \cdot x_k^2}$
ℓ_2-ℓ_2	$\frac{\lVert A_{i:}\rVert_1^2}{\sum_{k\in[m]} \lVert A_{k:}\rVert_1^2} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\frac{\lVert A_{:j}\rVert_1^2}{\sum_{k\in[n]} \lVert A_{:k}\rVert_1^2} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{:j}\rVert_1}$
ℓ_2-ℓ_2	$\frac{{y_i}^2}{\lVert y\rVert_2^2} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\frac{{x_j}^2}{\lVert x\rVert_2^2} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{:j}\rVert_1}$

Recall that the estimator is of the form $\tilde{g}(x,y) = \left(\frac{1}{p_{ij}} y_i A_{ij} \cdot e_j, -\frac{1}{q_{lk}} A_{lk} x_k \cdot e_l\right)$ where $i, j \sim p$ and $l, k \sim q$.

Table V
THE DISTRIBUTIONS p, q USED FOR OUR REDUCED VARIANCE COORDINATE GRADIENT ESTIMATOR

Setting	p_{ij}	q_{ij}
ℓ_1-ℓ_1	$\frac{y_i + 2[y_0]_i}{3} \cdot \frac{A_{ij}^2}{\lVert A_{i:}\rVert_2^2}$	$\frac{x_j + 2[x_0]_j}{3} \cdot \frac{A_{ij}^2}{\lVert A_{:j}\rVert_2^2}$
ℓ_2-ℓ_1	$\frac{y_i + 2[y_0]_i}{3} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\frac{A_{ij}^2}{\lVert A\rVert_{\mathrm{F}}^2}$
ℓ_2-ℓ_1	$\frac{y_i + 2[y_0]_i}{3} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\propto [x - x_0]_j^2 \cdot \mathbf{1}_{A_{ij}\neq 0}$
ℓ_2-ℓ_1	$\frac{y_i + 2[y_0]_i}{3} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\frac{\lvert A_{ij}\rvert \cdot [x - x_0]_j^2}{\sum_{k\in[n]} \lVert A_{:k}\rVert_1 \cdot [x - x_0]_k^2}$
ℓ_2-ℓ_2	$\frac{\lVert A_{i:}\rVert_1^2}{\sum_{k\in[m]} \lVert A_{k:}\rVert_1^2} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\frac{\lVert A_{:j}\rVert_1^2}{\sum_{k\in[n]} \lVert A_{:k}\rVert_1^2} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{:j}\rVert_1}$
ℓ_2-ℓ_2	$\frac{[y - y_0]_i^2}{\lVert y - y_0\rVert_2^2} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{i:}\rVert_1}$	$\frac{[x - x_0]_j^2}{\lVert x - x_0\rVert_2^2} \cdot \frac{\lvert A_{ij}\rvert}{\lVert A_{:j}\rVert_1}$

Recall that the estimator is of the form $\tilde{g}(x,y) = \left(A^\top y + \frac{1}{p_{ij}}(y_i - y_{0,i}) A_{ij} \cdot e_j, -Ax - \frac{1}{q_{lk}} A_{lk}(x_k - x_{0,k}) \cdot e_l\right)$ where $i, j \sim p$ and $l, k \sim q$ and x_0, y_0 is a reference point.

and break the recursion for the unnormalized iterates $\hat{x}_t$ into two steps

$$\hat{x}'_t = \hat{x}_t^{\kappa} \circ \exp\{v\}, \text{ and} \tag{9}$$

$$\hat{x}_{t+1} = \hat{x}'_t \circ \exp\{s_t\}, \tag{10}$$

where $v = (1-\kappa)\log x_0 - \eta\kappa g_0^{\mathsf{x}}$ is a fixed dense vector, and $s_t = -\eta\tilde{\delta}^{\mathsf{x}}(x_t, y_t)$ is a varying 1-sparse vector. The key task of the data structures is maintaining the normalization factor $\|\hat{x}_t\|_1$ in near-constant time. Standard data structures do not suffice because they lack support for the dense step (9).

Our high-level strategy is to handle the two steps (9) and (10) separately. To handle the dense step (9), we propose the data structure `ScaleMaintainer` that efficiently approximates $\|\hat{x}_t\|_1$ in the "homogeneous" case of no sparse updates (i.e. $s_t = 0$ for all t). We then add support for the sparse step (10) using a binomial heap-like construction involving $O(\log n)$ instances of `ScaleMaintainer`.

The `ScaleMaintainer` *data structure.* When $s_t = 0$ for all t the iterates $\hat{x}_t$ admit closed forms

$$\hat{x}_{t+\tau} = \hat{x}_t^{\kappa^\tau} \circ \exp\left\{ v \sum_{t'=0}^{\tau-1} \kappa^{t'} \right\} = \hat{x}_t^{\kappa^\tau} \circ \exp\left\{ \frac{1-\kappa^\tau}{1-\kappa} v \right\}$$
$$= \hat{x}_t \circ \exp\left\{ [1-\kappa^\tau]\bar{v} \right\},$$

where $\bar{v} = \frac{v}{1-\kappa} - \log x_t$. Consequently, we design `ScaleMaintainer` to take as initialization $\bar{n}$-dimensional vectors $\bar{x} \in \mathbb{R}^{\bar{n}}_{\geq 0}$, and $\bar{v} \in \mathbb{R}^{\bar{n}}$ and provide approximations of the normalization factor

$$Z_\tau(\bar{x}, \bar{v}) = \|\bar{x} \circ \exp\{(1-\kappa^\tau)\bar{v}\}\|_1 \tag{11}$$

for arbitrary values of $\tau \geq 1$. We show how to implement each query of $Z_\tau(\bar{x}, \bar{v})$ in amortized time $\widetilde{O}(1)$. The data structure also supports initialization in time $\widetilde{O}(\bar{n})$ and deletions (i.e., setting elements of $\bar{x}$ to zero) in amortized time $\widetilde{O}(1)$.

To efficiently approximate the quantity $Z_\tau(\bar{x}, \bar{v})$ we replace the exponential with its order $p = O(\log n)$ Taylor series. That is, we would like to write

$$Z_\tau(\bar{x}, \bar{v}) = \sum_{i \in [\bar{n}]} [\bar{x}]_i e^{(1-\kappa^\tau)[\bar{v}]_i}$$
$$\approx \sum_{i \in [\bar{n}]} [\bar{x}]_i \sum_{q=0}^{p} \frac{1}{q!} (1-\kappa^\tau)^q [\bar{v}]_i^q$$
$$= \sum_{q=0}^{p} \frac{(1-\kappa^\tau)^q}{q!} \langle \bar{x}, \bar{v}^q \rangle .$$

The approximation $\sum_{q=0}^{p} \frac{(1-\kappa^\tau)^q}{q!} \langle \bar{x}, \bar{v}^q \rangle$ is cheap to compute, since for every τ it is a linear combination of the $p = \widetilde{O}(1)$ numbers $\{\langle \bar{x}, \bar{v}^q \rangle\}_{q \in [p]}$ which we can compute once at initialization. However, the Taylor series approximation has low multiplicative error only when $|(1-\kappa^\tau)[\bar{v}]_i| = O(p)$, which may fail to hold, as we may have $\|\bar{v}\|_\infty = \mathsf{poly}(n)$ in general. To handle this, suppose that for a fixed τ we have an offset $\mu \in \mathbb{R}$ and "active set" $A \subseteq [\bar{n}]$ such that the following conditions hold for a threshold $R = O(p)$: (a) the Taylor approximation is valid in A, e.g. we have $|(1-\kappa^\tau)(\bar{v}_i - \mu)| \leq 2R$ for all $i \in A$, (b) entries outside A are small; $(1-\kappa^\tau)[\bar{v}_i - \mu] \leq -R$ for all $i \notin A$, and (c) at least one entry in the active set is large; $(1-\kappa^\tau)[\bar{v}_i - \mu] \geq 0$ for some $i \in A$. Under these conditions, the entries in A^c are negligibly small and we can truncate them, resulting in the approximation

$$e^{(1-\kappa^\tau)\mu} \left[\sum_{q=0}^{p} \frac{(1-\kappa^\tau)^q}{q!} \langle \bar{x}, (\bar{v}-\mu)^q \rangle_A + e^{-R} \langle \bar{x}, \mathbf{1} \rangle_{A^c} \right],$$

which we show approximates $Z_\tau(\bar{x}, \bar{v})$ to within $e^{O(R+\log n)-\Omega(p)}$ multiplicative error, where we used $\langle a, b \rangle_S := \sum_{i \in S} a_i b_i$; here, we also require that $\log \frac{\max_i \bar{x}_i}{\min_i \bar{x}_i} = O(R)$, which we guarantee when choosing the initial $\bar{x}$.

The challenge then becomes efficiently mapping any τ to $\{\langle \bar{x}, (\bar{v}-\mu)^q \rangle_A\}_{q \in [p]}$ for suitable μ and A. We address this by jointly bucketing τ and $\bar{v}$. Specifically, we map τ into a bucket index $k = \lfloor \log_2 \frac{1-\kappa^\tau}{1-\kappa} \rfloor$, pick μ to be the largest integer multiple of $R/((1-\kappa)2^k)$ such that $\mu \leq \max_i \bar{v}_i$, and set $A = \{i \mid |(1-\kappa)2^k(\bar{v}_i - \mu)| \leq R\}$. Since $k \leq k_{\max} = \lfloor \log_2 \frac{1}{1-\kappa} \rfloor = O(\log n)$, we argue that computing $\langle \bar{x}, (\bar{v}-\mu)^q \rangle_A$ for every possible resulting μ and A takes at most $O(\bar{n} p \log \frac{1}{1-\kappa}) = \widetilde{O}(\bar{n})$ time, which we can charge to initialization. We further show how to support deletions in $\widetilde{O}(1)$ time by carefully manipulating the computed quantities.

Supporting sparse updates. Building on `ScaleMaintainer`, we design the data structure `ApproxExpMaintainer` that (approximately) implements the entire mirror descent step (5) in time $\widetilde{O}(1)$.[2] The data structure maintains vectors $\bar{x} \in \Delta^n$ and $\bar{v} \in \mathbb{R}^n$ and $K = \lceil \log_2(n+1) \rceil$ instances of `ScaleMaintainer` denoted $\{\texttt{ScaleMaintainer}_k\}_{k \in [K]}$. The kth instance tracks a coordinate subset $S_k \subseteq [n]$ such that $\{S_k\}_{k \in [K]}$ partitions $[n]$, and has initial data $[\bar{x}]_{S_k}$ and $[\bar{v}]_{S_k}$. We let $\tau_k \geq 0$ denote the "time index" parameter of the kth instance. The data structure satisfies two invariants; first, the unnormalized iterate $\hat{x}$ satisfies

$$[\hat{x}]_{S_k} = [\bar{x} \circ \exp\{(1-\kappa^{\tau_k})\bar{v}\}]_{S_k}, \text{ for all } k \in [K]. \tag{12}$$

Second, the partition satisfies

$$|S_k| \leq 2^k - 1 \text{ for all } k \in [K], \tag{13}$$

[2]The data structures `ApproxExpMaintainer` and `ScaleMaintainer` structure support two additional operations necessary for our algorithm: efficient approximate sampling from x_t and maintenance of a running sum of $\hat{x}_\tau$. Given the normalization constant approximation, the implementation of these operations is fairly straightforward, so we do not discuss them in the introduction.

where at initialization we let $S_K = [n]$ and $S_k = \emptyset$ for $k < K$, $\bar{x} = x_0$, $\bar{v} = \frac{v}{1-\kappa} - \log x_0$ and $\tau_K = 0$.

The invariant (12) allows us to efficiently (in time $\widetilde{O}(K) = \widetilde{O}(1)$) query coordinates of $x_t = \hat{x}_t / \|\hat{x}_t\|_1$, since ScaleMaintainer allows us to approximate $\|\hat{x}_t\|_1 = \sum_{k\in[K]} Z_{\tau_k}([\bar{x}]_{S_k}, [\bar{v}]_{S_k})$ with Z as defined in (11). To implement the dense step (9), we simply increment $\tau_k \leftarrow \tau_k + 1$ for every k. Let j be the nonzero coordinate of s_t in the sparse step (10), and let $k \in [K]$ be such that $j \in S_k$. To implement (10), we delete coordinate j from $\texttt{ScaleMaintainer}_k$, and create a singleton instance $\texttt{ScaleMaintainer}_0$ maintaining $S_0 = \{j\}$ with initial data $[\bar{x}]_{S_0} = e^{s_t}\hat{x}_j$, $[\bar{v}]_{S_0} = v_j/(1-\kappa) - \log(e^{s_t}\hat{x}_j)$ and $\tau_0 = 0$. Going from $k = 1$ to $k = K$, we merge $\texttt{ScaleMaintainer}_{k-1}$ into $\texttt{ScaleMaintainer}_k$ until the invariant (13) holds again. For example, if before the sparse step we have $|S_1| = 1$, $|S_2| = 3$ and $|S_3| = 2$, we will perform 3 consecutive merges, so that afterwards we have $|S_1| = |S_2| = 0$ and $|S_3| = 7$.

To merge two $\texttt{ScaleMaintainer}_{k-1}$ into $\texttt{ScaleMaintainer}_k$, we let $S'_k = S_{k-1} \cup S_k$ and initialize a new ScaleMaintainer instance with $[\bar{x}]_{S'_k} = [\hat{x}]_{S'_k}$,[3] $[\bar{v}]_{S'_k} = [v]_{S'_k}/(1-\kappa) - \log[\hat{x}]_{S'_k}$ and $\tau_k = 0$; this takes $\widetilde{O}(|S'_k|) = \widetilde{O}(2^k)$ time due to the invariant (13). Noting that a merge at level k can only happen once in every $\Omega(2^k)$ updates, we conclude that the amortized cost of merges at each level is $\widetilde{O}(1)$, and (since $K = \widetilde{O}(1)$), so is the cost of the sparse update.

Back to distribution design (sampling from the sum). Our data structure enables us to compute the iteration (5) and query coordinates of the iterates x_t and y_t in $\widetilde{O}(1)$ amortized time. However, we cannot compute $\tilde{\delta}^{\mathsf{x}}$ using the distribution (8) because we do not have an efficient way of sampling from $|y_t - y_0|$; Taylor approximation techniques are not effective for approximating the absolute value because it is not smooth. To overcome this final barrier, we introduce a new design which we call "sampling from the sum,"

$$p_{ij}(x,y) = \left(\frac{1}{3}y_i + \frac{2}{3}[y_0]_i\right) \cdot \frac{A_{ij}^2}{\|A_{i:}\|_2^2}. \qquad (14)$$

Sampling from the modified distribution is simple, as our data structure allows us to sample from y_t. Moreover, we show that the distribution (14) satisfies a relaxed version of (6) where the LHS is replaced by a local norm as before, and the RHS is replaced by $L^2(V_{x_0}(x_t) + V_{y_0}(y_t))$, where $V_x(x')$ is the KL divergence between x and x'. In Table V we list the sampling distributions we design for variance reduction in the different domain geometries.

[3]More precisely, for every $j \in S'_k$ we set $\bar{x}_j = \hat{x}_j + \varepsilon \max_{i \in S'_k} \hat{x}_i$, where ε is a small padding constant that ensures the bounded multiplicative range necessary for correct operation of ScaleMaintainer.

C. Related work

Coordinate methods. Updating a single coordinate at a time—or more broadly computing only a single coordinate of the gradient at every iteration—is a well-studied and successful technique in optimization [23]. Selecting coordinates at random is key to obtaining strong performance guarantees: Strohmer and Vershynin [24] show this for linear regression, Shalev-Shwartz and Tewari [25] show this for ℓ_1 regularized linear models, and Nesterov [26] shows this for general smooth minimization. Later works [27–29] propose accelerated coordinate methods. These works share two common themes: selecting the gradient coordinate from a non-uniform distribution (see also [30]), and augmenting the 1-sparse stochastic gradient with a dense momentum term. These techniques play important roles in our development as well.

To reap the full benefits of coordinate methods, iterations must be very cheap, ideally taking near-constant time. However, most coordinate methods require super-constant time, typically in the form of a vector-vector computation. Even works that consider coordinate methods in a primal-dual context [12, 22, 31–33] perform the coordinate updates only on the dual variable and require a vector-vector product (or more generally a component gradient computation) at every iteration.

A notable exception is the work of Wang [34, 35] which develops a primal-dual stochastic coordinate method for solving Markov decision processes, essentially viewing them as ℓ_∞-ℓ_1 bilinear saddle-point problems. Using a tree-based ℓ_1 sampler data structure similar to the ℓ_1 sampler we use for simplex domains for the sublinear case, the method allows for $\widetilde{O}(1)$ iterations and a potentially sublinear runtime scaling as ϵ^{-2}. Tan et al. [36] also consider bilinear saddle-point problems and variance reduction. Unlike our work, they assume a separable domain, use uniform sampling, and do not accelerate their variance reduction scheme with extra-gradient steps. The separable domain makes attaining constant iteration cost time much simpler, since there is no longer a normalization factor to track, but it also rules out applications to the simplex domain. While Tan et al. [36] report promising empirical results, their theoretical guarantees do not improve upon prior work.

Our work develops coordinate methods with $\widetilde{O}(1)$ iteration cost for new types of problems. Furthermore, it maintains the iteration efficiency even in the presence of dense components arising from the update, thus allowing for acceleration via an extra-gradient scheme.

Data structures for optimization. Performing iterations in time that is asymptotically smaller than the number of variables updated at every iteration forces us to carry out the updates implicitly using data structures; several prior works employ data structures for exactly the same reason. One of the most similar examples comes from Lee and

Sidford [27], who design a data structure for an accelerated coordinate method in Euclidean geometry. In our terminology, their data structure allows performing each iteration in time $O(\mathsf{rcs})$ while implicitly updating variables of size $O(n)$. Duchi et al. [37] design a data structure based on balanced search trees that supports efficient Euclidean projection to the ℓ_1 ball of vector of the form $u + s$ where u is in the ℓ_1 ball and s is sparse. They apply it in a stochastic gradient method for learning ℓ_1 regularized linear classifier with sparse features. Among the many applications of this data structure, Namkoong and Duchi [33] adapt it to efficiently compute Euclidean projections into the intersection of the simplex and a χ^2 ball for 1-sparse updates. Shalev-Shwartz and Wexler [22] and Wang [34, 35], among others, use binary tree data structures to perform multiplicative weights projection to the simplex and sampling from the iterates.

A recent work of Sidford and Tian [38] develops a data structure which is somewhat similar to our `ApproxExpMaintainer` data structure, for updates arising from a primal-dual method to efficiently solve ℓ_∞ regression. Their data structure was also designed to handle updates to a simplex variable which summed a structured dense component and a sparse component. However, the data structure design of that work specifically exploited the structure of the maximum flow problem in a number of ways, such as bounding the sizes of the update components and relating these bounds to how often the entire data structure should be restarted. Our data structure can handle a broader range of structured updates to simplex variables and has a much more flexible interface, which is crucial to the development of our variance-reduced methods as well as our applications.

Another notable use of data structures in optimization appears in second order methods, where a long line of work uses them to efficiently solve sequences of linear systems and approximately compute iterates [14–16, 39–42]. Finally, several works on low rank optimization make use of sketches to efficiently represent their iterates and solutions [43, 44].

Numerical sparsity. Measures of numerical sparsity, such as the ℓ_2/ℓ_∞ or ℓ_1/ℓ_2 ratios, are continuous and dimensionless relaxations of the ℓ_0 norm. The *stable rank* of a matrix A measures the numerical sparsity of its singular values (specifically, their squared ℓ_2/ℓ_∞ ratio) [45].

For linear regression, stochastic methods generally outperform exact gradient methods only when A is has low stable rank, cf. discussion in [11, Section 4.3], i.e., numerically sparse singular values. In recent work, Gupta and Sidford [6] develop algorithms for linear regression and eigenvector problems for matrices with numerically sparse entries (as opposed to singular values). Our paper further broadens the scope of matrix problems for which we can benefit from numerical sparsity. Moreover, our results have implications for regression as well, improving on [6] in certain numerically sparse regimes.

In recent work by Babichev et al. [46], the authors develop primal-dual sublinear methods for ℓ_1-regularized linear multi-class classification (bilinear games in ℓ_1-ℓ_∞ geometry), and obtain complexity improvements depending on the numerical sparsity of the problem. Similarly to our work, careful design of the sampling distribution plays a central role in [46]. They also develop a data structure that allows iteration cost independent of the number of classes. However, unlike our work, Babichev et al. [46] rely on sampling entire rows and columns, have iteration costs linear in $n + m$, and do not utilize variance reduction. We believe that our techniques can yield improvements in their setting.

Acknowledgements

This research was supported in part by Stanford Graduate Fellowships, NSF CAREER Award CCF-1844855, NSF Graduate Fellowship DGE-1656518 and a PayPal research gift. We thank the anonymous reviewers who helped improve the completeness and readability of this paper by providing many helpful comments.

References

[1] Y. Carmon, Y. Jin, A. Sidford, and K. Tian, "Coordinate methods for matrix games," *arXiv preprint arXiv:2009.08447*, 2020.

[2] M. Minsky and S. Papert, *Perceptrons—an introduction to computational geometry*. MIT Press, 1987.

[3] J. Von Neumann and O. Morgenstern, *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 1944.

[4] G. B. Dantzig, *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press, 1953.

[5] K. L. Clarkson, E. Hazan, and D. P. Woodruff, "Sublinear optimization for machine learning," in *51th Annual IEEE Symposium on Foundations of Computer Science*, 2010, pp. 449–457.

[6] N. Gupta and A. Sidford, "Exploiting numerical sparsity for efficient learning: faster eigenvector computation and regression," in *Advances in Neural Information Processing Systems*, 2018, pp. 5269–5278.

[7] A. Nemirovski, "Prox-method with rate of convergence $O(1/t)$ for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems," *SIAM Journal on Optimization*, vol. 15, no. 1, pp. 229–251, 2004.

[8] Y. Nesterov, "Dual extrapolation and its applications to solving variational inequalities and related problems," *Mathematical Programing*, vol. 109, no. 2-3, pp. 319–344, 2007.

[9] M. D. Grigoriadis and L. G. Khachiyan, "A sublinear-time randomized approximation algorithm for matrix games," *Operation Research Letters*, vol. 18, no. 2, pp. 53–58, 1995.

[10] P. Balamurugan and F. R. Bach, "Stochastic variance reduction methods for saddle-point problems," in *Advances in Neural Information Processing Systems*, 2016.

[11] Y. Carmon, Y. Jin, A. Sidford, and K. Tian, "Variance reduction for matrix games," in *Advances in Neural Information Processing Systems*, 2019.

[12] Z. Allen-Zhu, Z. Liao, and Y. Yuan, "Optimization algorithms for faster computational geometry," in *43rd International Colloquium on Automata, Languages, and Programming*, 2016, pp. 53:1–53:6.

[13] Y. Nesterov, "A method for solving a convex programming problem with convergence rate $o(1/k^2)$," *Doklady AN SSSR*, vol. 269, pp. 543–547, 1983.

[14] Y. T. Lee and A. Sidford, "Efficient inverse maintenance and faster algorithms for linear programming," in *IEEE 56th Annual Symposium on Foundations of Computer Science*, 2015, pp. 230–249.

[15] M. B. Cohen, Y. T. Lee, and Z. Song, "Solving linear programs in the current matrix multiplication time," *arXiv preprint arXiv:1810.07896*, 2018.

[16] J. van den Brand, Y. T. Lee, A. Sidford, and Z. Song, "Solving tall dense linear programs in nearly linear time," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020, to appear.

[17] M. G. Azar, R. Munos, and H. J. Kappen, "Minimax pac bounds on the sample complexity of reinforcement learning with a generative model," *Machine learning*, vol. 91, no. 3, pp. 325–349, 2013.

[18] A. Sidford, M. Wang, X. Wu, and Y. Ye, "Variance reduced value iteration and faster algorithms for solving markov decision processes," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2018, pp. 770–787.

[19] A. Gilyén, S. Lloyd, and E. Tang, "Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension," *arXiv preprint arXiv:1811.04909*, 2018.

[20] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[21] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[22] S. Shalev-Shwartz and Y. Wexler, "Minimizing the maximal loss: How and why." in *ICML*, 2016, pp. 793–801.

[23] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[24] T. Strohmer and R. Vershynin, "A randomized Kaczmarz algorithm with exponential convergence," *Journal of Fourier Analysis and Applications*, vol. 15, no. 2, p. 262, 2009.

[25] S. Shalev-Shwartz and A. Tewari, "Stochastic methods for ℓ_1-regularized loss minimization," *Journal of Machine Learning Research*, vol. 12, pp. 1865–1892, 2011.

[26] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[27] Y. T. Lee and A. Sidford, "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 2013.

[28] Z. Allen-Zhu, Z. Qu, P. Richtárik, and Y. Yuan, "Even faster accelerated coordinate descent using non-uniform sampling," in *International Conference on Machine Learning*, 2016, pp. 1110–1119.

[29] Y. Nesterov and S. U. Stich, "Efficiency of the accelerated coordinate descent method on structured optimization problems," *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 110–123, 2017.

[30] P. Richtárik and M. Takáč, "On optimal probabilities in stochastic coordinate descent methods," *Optimization Letters*, vol. 10, no. 6, pp. 1233–1243, 2016.

[31] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, pp. 567–599, 2013.

[32] Y. Zhang and L. Xiao, "Stochastic primal-dual coordinate method for regularized empirical risk minimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2939–2980, 2017.

[33] H. Namkoong and J. C. Duchi, "Stochastic gradient methods for distributionally robust optimization with f-divergences," in *Advances in Neural Information Processing Systems*, 2016, pp. 2208–2216.

[34] M. Wang, "Randomized linear programming solves the discounted Markov decision problem in nearly-linear (sometimes sublinear) running time," *arXiv preprint arXiv:1704.01869*, 2017.

[35] ——, "Primal-dual π learning: Sample complexity and sublinear run time for ergodic Markov decision problems," *arXiv preprint arXiv:1710.06100*, 2017.

[36] C. Tan, T. Zhang, S. Ma, and J. Liu, "Stochastic primal-dual method for empirical risk minimization with $o(1)$ per-iteration complexity," in *Advances in Neural Information Processing Systems*, 2018.

[37] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l 1-ball for learning in high dimensions," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 272–279.

[38] A. Sidford and K. Tian, "Coordinate methods for accelerating ℓ_∞ regression and faster approximate maximum flow," in *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris,*

France, October 7-9, 2018, 2018, pp. 922–933.

[39] N. Karmarkar, “A new polynomial-time algorithm for linear programming,” in *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM, 1984, pp. 302–311.

[40] E. Andersen, C. Roos, T. Terlaky, T. Trafalis, and J. Warners, “The use of low-rank updates in interior-point methods,” *Numerical Linear Algebra and Optimization*, pp. 1–12, 1996.

[41] Y. T. Lee, Z. Song, and Q. Zhang, “Solving empirical risk minimization in the current matrix multiplication time,” in *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, 2019, pp. 2140–2157.

[42] J. van den Brand, “A deterministic linear program solver in current matrix multiplication time,” *arXiv preprint arXiv:1910.11957*, 2019.

[43] K. L. Clarkson and D. P. Woodruff, “Low rank approximation and regression in input sparsity time,” in *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*. ACM, 2013, pp. 81–90.

[44] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher, “Sketchy decisions: Convex low-rank matrix optimization with optimal storage,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1188–1196.

[45] M. B. Cohen, J. Nelson, and D. P. Woodruff, “Optimal approximate matrix product in terms of stable rank,” in *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[46] D. Babichev, D. Ostrovskii, and F. Bach, “Efficient primal-dual algorithms for large-scale multiclass classification,” *arXiv preprint arXiv:1902.03755*, 2019.