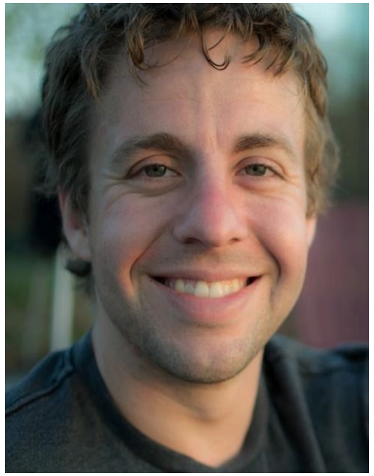
Fast, Parallel, Query-Efficient Binary Classification



Ishani Karmarkar

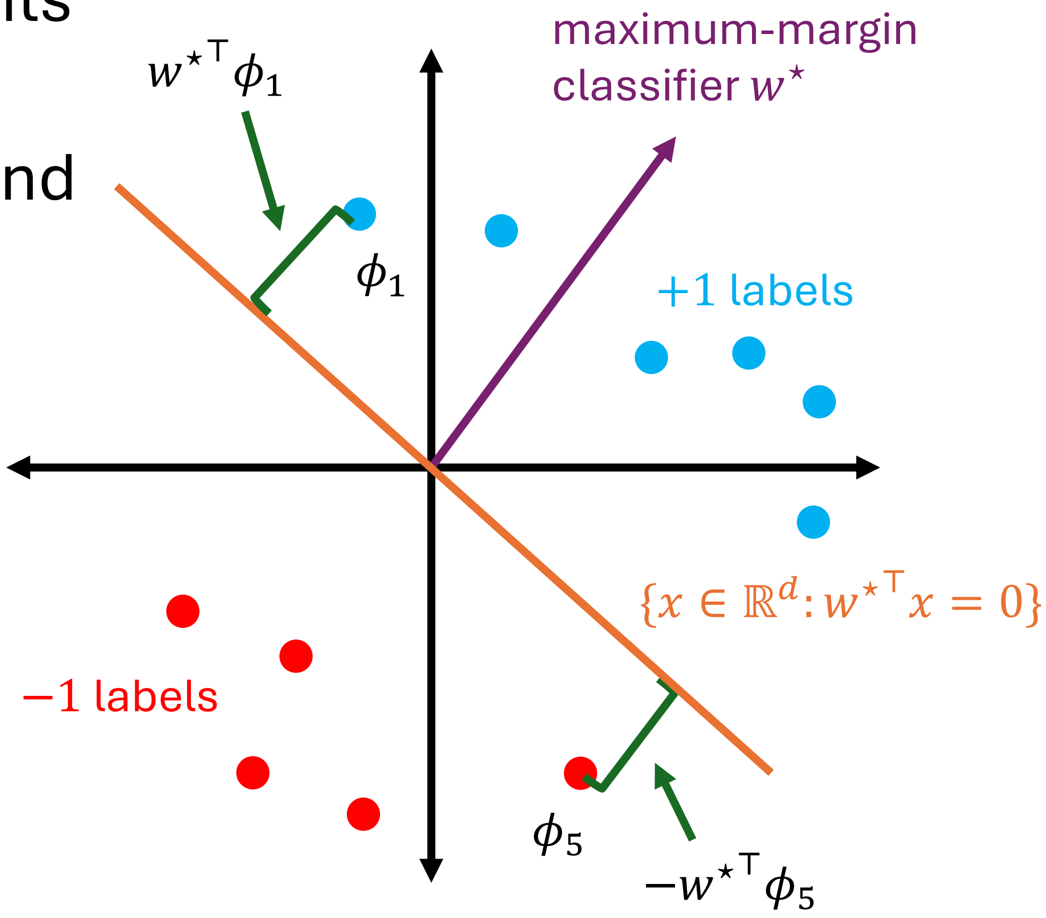
Liam O'Carroll

Aaron Sidford

Stanford University

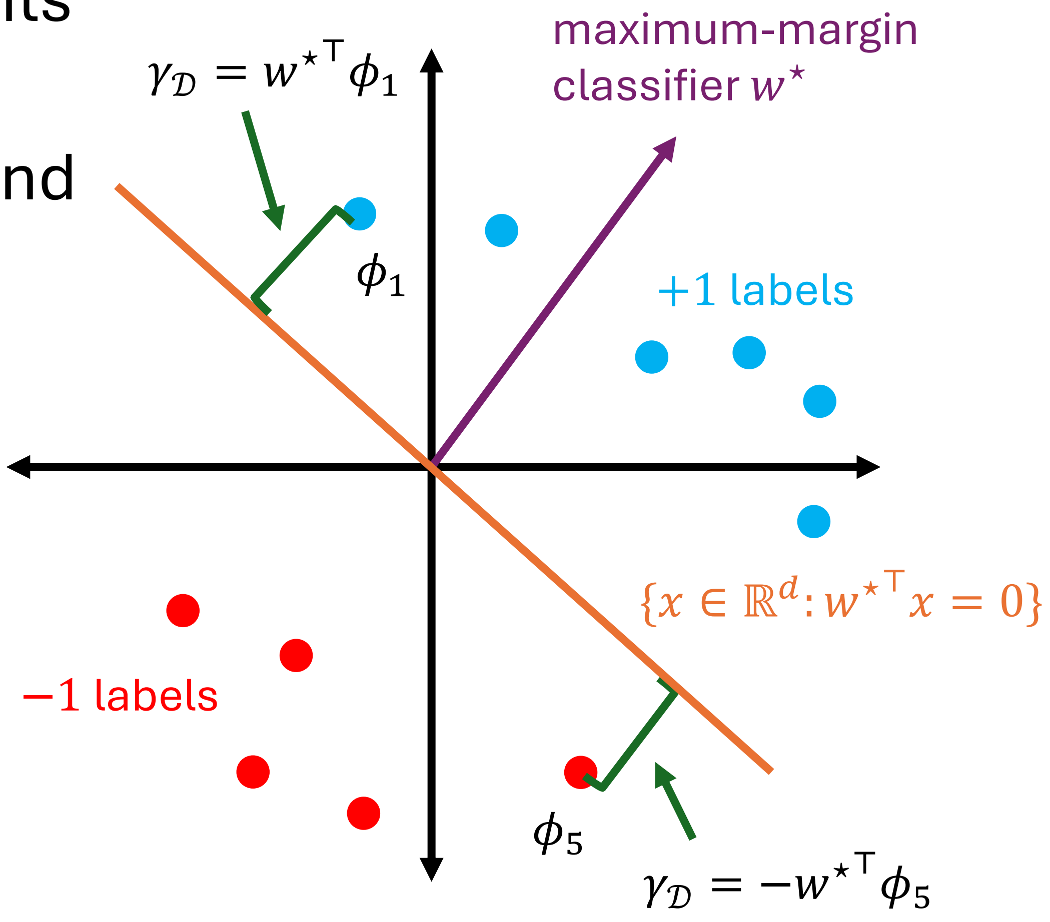
Maximum-margin binary classification

- Foundational binary data classification problem: find a separating hyperplane for two sets of points
- **Setup:** Given dataset $\mathcal{D} = (\phi_1, l_1), \dots, (\phi_n, l_n)$ where $\phi_i \in \mathbb{B}^d$ are normalized feature vectors and $l_i \in \{+1, -1\}$ are binary labels. $\mathbb{B}^d = \{x \in \mathbb{R}^d : \|x\|_2 \leq 1\}$ denotes the unit ball.



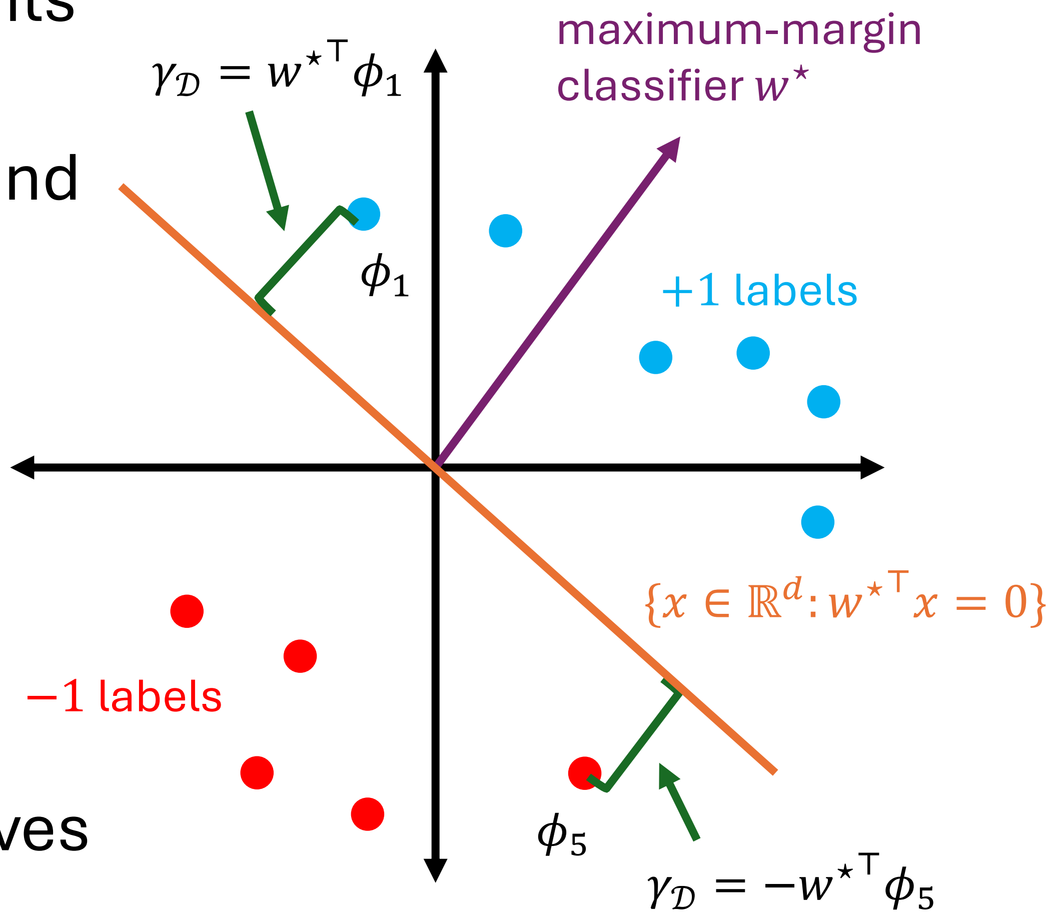
Maximum-margin binary classification

- Foundational binary data classification problem: find a separating hyperplane for two sets of points
- **Setup:** Given dataset $\mathcal{D} = (\phi_1, l_1), \ldots, (\phi_n, l_n)$ where $\phi_i \in \mathbb{B}^d$ are normalized feature vectors and $l_i \in \{+1, -1\}$ are binary labels. $\mathbb{B}^d = \{x \in \mathbb{R}^d : \|x\|_2 \leq 1\}$ denotes the unit ball.



Maximum-margin binary classification

- Foundational binary data classification problem: find a separating hyperplane for two sets of points
- **Setup:** Given dataset $\mathcal{D} = (\phi_1, l_1), \dots, (\phi_n, l_n)$ where $\phi_i \in \mathbb{B}^d$ are normalized feature vectors and $l_i \in \{+1, -1\}$ are binary labels. $\mathbb{B}^d = \{x \in \mathbb{R}^d : \|x\|_2 \leq 1\}$ denotes the unit ball.
- The *maximum margin* $\gamma_{\mathcal{D}}$ of the dataset is
$$\gamma_{\mathcal{D}} = \max_{w \in \mathbb{B}^d} \min_{i \in [n]} l_i \cdot w^\top \phi_i$$
and thus there exists $w^\star \in \mathbb{B}^d$ such that
$$l_i \cdot {w^\star}^\top \phi_i \geq \gamma_{\mathcal{D}}, \qquad \forall i \in [n]$$
- **Goal:** Given $\rho > 0$, output $\hat{w} \in \mathbb{B}^d$ which achieves a margin of at least $\gamma_{\mathcal{D}} - \rho$, in other words:
$$l_i \cdot \hat{w}^\top \phi_i \geq \gamma_{\mathcal{D}} - \rho, \qquad \forall i \in [n]$$



Maximum-margin binary classification

- Foundational binary data classification problem: find a separating hyperplane for two sets of points
- **Setup:** Given dataset $\mathcal{D} = (\phi_1, l_1), \ldots, (\phi_n, l_n)$ where $\phi_i \in \mathbb{B}^d$ are normalized feature vectors and $l_i \in \{+1, -1\}$ are binary labels. $\mathbb{B}^d = \{x \in \mathbb{R}^d : \|x\|_2 \leq 1\}$ denotes the unit ball.
- The *maximum margin* $\gamma_{\mathcal{D}}$ of the dataset is

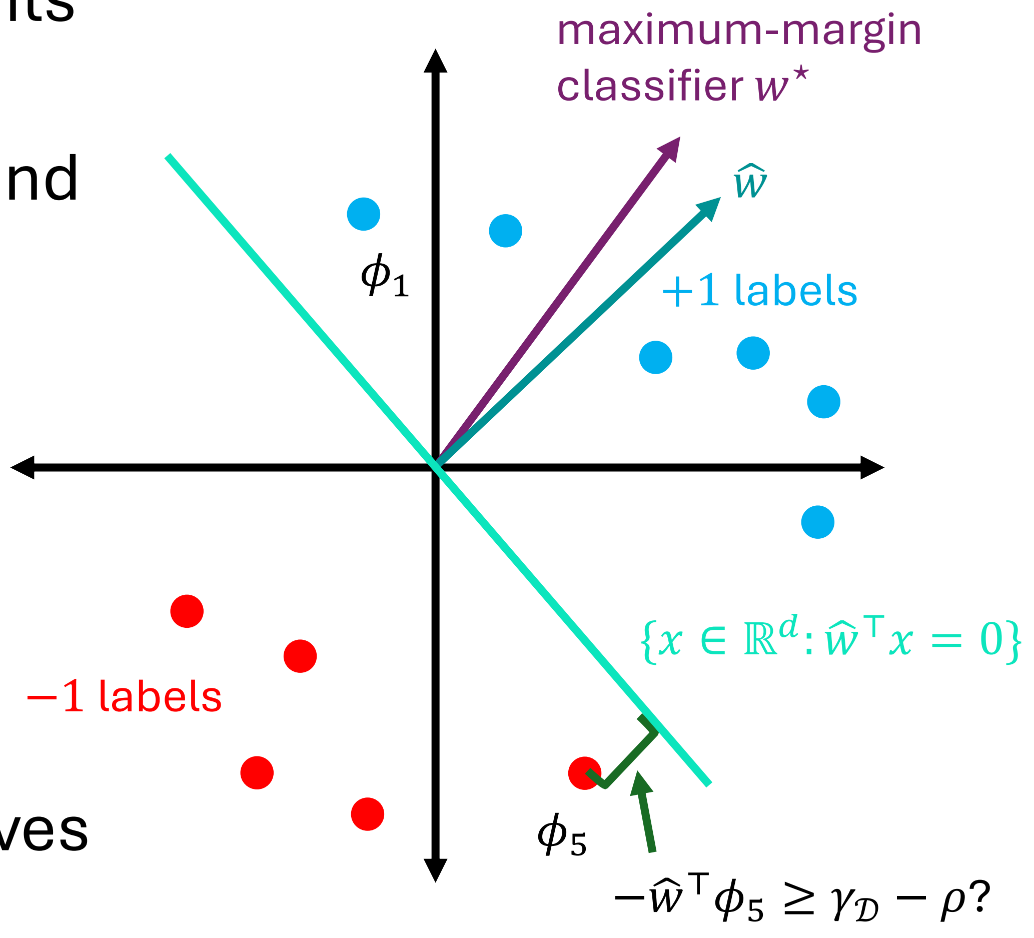
$$\gamma_{\mathcal{D}} = \max_{w \in \mathbb{B}^d} \min_{i \in [n]} l_i \cdot w^\top \phi_i$$

  and thus there exists $w^\star \in \mathbb{B}^d$ such that

$$l_i \cdot {w^\star}^\top \phi_i \geq \gamma_{\mathcal{D}}, \qquad \forall i \in [n]$$

- **Goal:** Given $\rho > 0$, output $\widehat{w} \in \mathbb{B}^d$ which achieves a margin of at least $\gamma_{\mathcal{D}} - \rho$, in other words:

$$l_i \cdot \widehat{w}^\top \phi_i \geq \gamma_{\mathcal{D}} - \rho, \qquad \forall i \in [n]$$



Complexity metrics and motivation

We study three complexity metrics:

- **Work:** Total sequential runtime
  - E.g., adding two vectors in $\mathbb{R}^k$ takes $O(k)$ work
- **Depth:** Parallel runtime
  - E.g., adding two vectors in $\mathbb{R}^k$ takes $O(1)$ depth
- **Matvec complexity:** Our algorithms *only* access the dataset through matrix-vector multiplies to the label-signed design matrix $\Phi_{\mathcal{D}}$. Matvec complexity is the number of such queries.
  - Formally, a matvec query is $\left(\Phi_{\mathcal{D}}^{\top} y, \Phi_{\mathcal{D}} x\right)$ for vectors x, y, where $\Phi_{\mathcal{D}} \in \mathbb{R}^{n \times d}$ denotes the matrix whose i-th row is $l_i \phi_i^{\top}$.

Key motivation: It was recently shown [KS25a, KOS26] that $\widetilde{\Theta}\left(\rho^{-2/3}\right)$ matvecs are necessary and sufficient for deterministic algorithms. However, the algorithm [KOS26] was not computationally efficient.

Question: Can we match the optimal $\widetilde{\Theta}\left(\rho^{-2/3}\right)$ matvec complexity while simultaneously obtaining work and depth improvements?

Our results

$\omega \approx 2.37$ is matrix multiplication constant.
$\mathrm{nnz}(\cdot)$ is number of nonzero entries (sparsity).
We hide polylog factors in the tables for brevity.

We answer in the affirmative, giving two fast, parallel, query-efficient algorithms.

- **Theorem 1 (More parallel):** There is a very parallel randomized algorithm achieving the complexities in the first row of the table.
- **Theorem 2 (Faster work):** There is an even faster randomized algorithm (albeit with increased depth) achieving the complexities in the second row.

Algorithm	Work	Depth	Matvecs
More parallel	$\mathrm{nnz}(\Phi_{\mathcal{D}})\rho^{-2/3} + \rho^{-2(\omega+1)/3}$	$\rho^{-2/3}$	$\rho^{-2/3}$
Faster work	$\mathrm{nnz}(\Phi_{\mathcal{D}})\rho^{-2/3} + \rho^{-2}$	$\rho^{-4/3}$	$\rho^{-2/3}$

Comparison to prior work

$\omega \approx 2.37$ is matrix multiplication constant.
$\mathrm{nnz}(\cdot)$ is number of nonzero entries (sparsity).
We hide polylog factors in the tables for brevity.

Algorithm	Work	Depth	Matvecs
More parallel	$\mathrm{nnz}(\Phi_{\mathcal{D}})\rho^{-2/3} + \rho^{-2(\omega+1)/3}$	$\rho^{-2/3}$	$\rho^{-2/3}$
Faster work	$\mathrm{nnz}(\Phi_{\mathcal{D}})\rho^{-2/3} + \rho^{-2}$	$\rho^{-4/3}$	$\rho^{-2/3}$

Table 1: Complexities achieved by our two algorithms

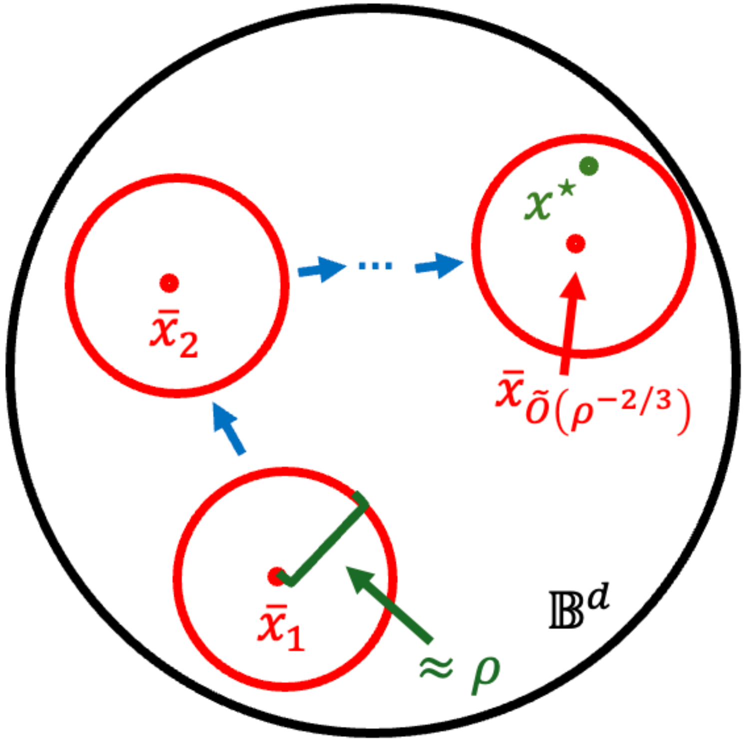
Algorithm	Work
(Exact) gradient methods [Nem04, Nes07]	$\mathrm{nnz}(\Phi_{\mathcal{D}})\rho^{-1}$
Row-col randomized method [GK95, PB16, CHW12]	$(n+d)\rho^{-2}$
Row-col randomized method with variance-reduction [PB16, CJST19]	$\mathrm{nnz}(\Phi_{\mathcal{D}}) + \sqrt{\mathrm{nnz}(\Phi_{\mathcal{D}}) \cdot (n+d)}\rho^{-1}$
Primal ball-accelerated stochastic methods [CJJS24]	$nd + nd^{2/3}\rho^{-2/3} + d\rho^{-2}$
Faster work	$\mathrm{nnz}(\Phi_{\mathcal{D}})\rho^{-2/3} + \rho^{-2}$

Table 2: Comparison of our faster work algorithm to the prior art.

- **Both algorithms** match the optimal (deterministic) $\widetilde{\Theta}(\rho^{-2/3})$ matvec complexity [KS25a, KOS26] while giving first simultaneous work and depth improvements
- **More parallel** algorithm improves upon previous best depth of $\tilde{O}(\rho^{-1})$ achieved by AGD, mirror prox, dual extrapolation, …
- **Faster work** algorithm is faster than all prior algorithms in certain regimes, e.g., sparse regimes where $\mathrm{nnz}(\Phi_{\mathcal{D}}) \approx n + d$ in the low-to-moderate precision range that includes statistically natural accuracies like $\rho \approx 1/\sqrt{n}$

Brief overview of techniques

- **Our main technical contribution:** We show how to carefully combine recent advanced techniques from several areas to achieve improved complexities, including:
  - ball-oracle acceleration methods [CJJ+20, CJJS21, ACJ+21a, CHJ+22, CJJ+23, CJJS24]
  - modern numerical linear algebra techniques, including oblivious random subspace embeddings [DS26, CDD24, CDDR24, CNW15, Coh16, NN13]
  - sample reuse framework/pseudo-independent algorithms [JKSW26]
- **Overview of algorithmic framework:**
  - **Part 1:** Reduce to solving a sequence of $\tilde{O}(\rho^{-2/3})$ local regularized subproblems. Further reduce each subproblem to solving $\tilde{O}(1)$ linear systems.
  - **Part 2:** Carefully apply modern numerical linear algebra techniques to solve sequence of linear systems efficiently. Key trick: Sample reuse framework [JKSW26] allows us to reuse a single fixed random subspace embedding over all $\tilde{O}(\rho^{-2/3})$ iterations.



Takeaway and open problems

- **Key takeaway:** We give two algorithms which simultaneously achieve the optimal (deterministic) $\widetilde{\Theta}(\rho^{-2/3})$ matvec complexity as well as work and depth improvements.

Algorithm	Work	Depth	Matvecs
More parallel	$\mathrm{nnz}(\Phi_{\mathcal{D}})\rho^{-2/3} + \rho^{-2(\omega+1)/3}$	$\rho^{-2/3}$	$\rho^{-2/3}$
Faster work	$\mathrm{nnz}(\Phi_{\mathcal{D}})\rho^{-2/3} + \rho^{-2}$	$\rho^{-4/3}$	$\rho^{-2/3}$

- **Open problems:**
  - Can we get the best depth (of more parallel algorithm) and best work (of faster work algorithm) simultaneously?
  - Can the guarantees be derandomized?
  - Can we extend to other related minimax optimization problems, such as ℓ_1-ℓ_1 matrix games (aka computing a Nash equilibrium of a zero-sum game)?