The Structural Complexity of Matrix Vector Multiplication

Emile Anand[1], Jan van den Brand[1], Rose McCarty[1,2]
[1] Georgia Tech [2] Princeton University

Model 1: Static Matrix-Vector Multiplication

- Given an $n \times n$ matrix $\mathbf{M}$, *preprocess* it,
- Support queries that for any vector v, returns $\mathbf{M}v$.

Model 2: Dynamic Matrix-Vector Multiplication

- Given an $n \times n$ matrix $\mathbf{M}$, preprocess it.
- Support the following queries:
  - For any vector v, return $\mathbf{M}v$
  - Update any row or column of the matrix

Can we perform queries in less than $O(n^2)$ operations?
- Computing matrix-vector products is **essential** in optimization, computational geometry, online/dynamic algorithms (e.g. neural network inference and backpropagation)
- Any complexity improvement has *wide-ranging implications*, and is thus a prevalent research topic in both **theory** and **practice**.

Substantial theoretical lower bounds:
1) Any poly(n)-space $\mathbf{M}v$ algorithm over sufficiently large finite fields needs $\Omega(n^2/\log n)$ time (Gronlund, Larsen, 2015),
2) $\Omega(n^2/\log n)$ for arithmetic circuits (Frandsen et. al., 2001),
3) Also for average case matrices (Henzinger et. al., 2022)

The **only** non-trivial upper bounds are over the Boolean semiring where $x \oplus y = min(1, x+y)$. Here, $\mathbf{M}v$ can be done in time $O(n^{2-o(1)})$ (Williams 2007, Abboud et. al. 2024).

Hardness conjecture (OMv hypothesis) Even over the Boolean semiring, no truly subquadratic time algorithm exists.

Success of Practical Heuristics Tremendous progress in $\mathbf{M}v$ multiplication through heuristics that run in nnz($\mathbf{M}$) worst-case time, but are much faster in practice (Alves et. al. 2024).

Beyond average-case analysis. Since the average-case (i.e., random non-structured input) is hard, the efficiency of practical algorithms must stem from inherent structure in real-world data.

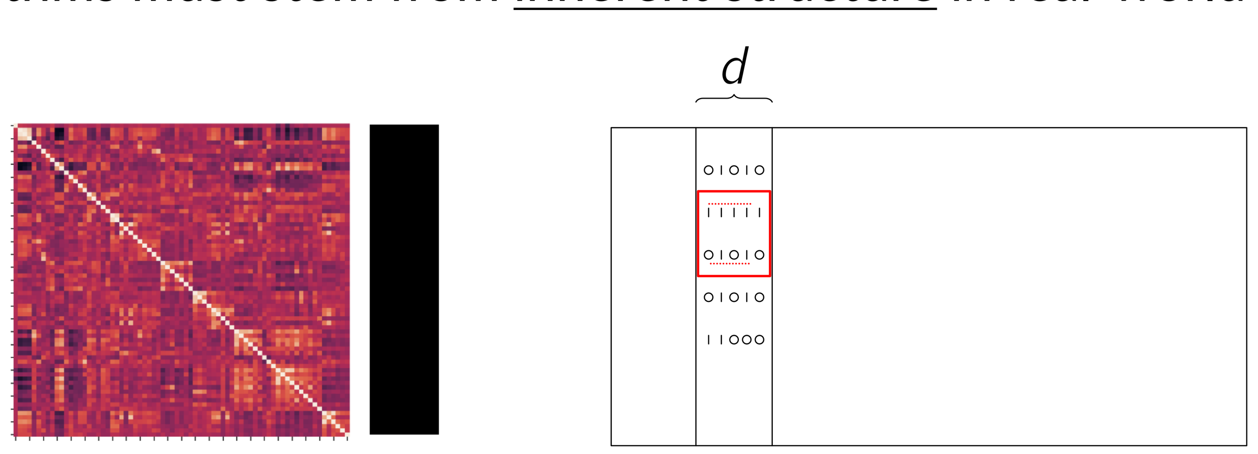

Figure 1 a) Structured $\mathbf{M}v$, b) low VC-dimension matrix

VC-dimension parameterizes the structural complexity of a Boolean matrix. $\mathbf{M}$ has VC-dimension d if the largest subset of columns S that contains every string in $\{0,1\}^{|S|}$ has size d.

Corrupted VC-dimension. A set system $\mathcal{F}$ on a ground set of size n has corrupted VC-dimension d if there is another set system $\mathcal{F}'$ of VC-dimension $\leq d$ such that $\mathcal{F}$ can be obtained from $\mathcal{F}'$ by adding/removing at most $O(n^{1-1/d})$ elements to or from each set.

If the corrupted VC-dimension of $\mathbf{M} \in \mathbb{Z}^{n\times n}$ is d, we can **preprocess M in $\tilde{O}(n^2)$ time** such that **$\mathbf{M}v$ can be computed in time $O(n^{2-1/d})$.**

Real-world data has constant VC-dimension.
Coudert et. al. (2024) computed the VC-dimension of families of real-world graph. Even for graphs with millions of nodes, the VC-dimension was between 3 to 8.

Matrices with constant VC-dimension:
- Adjacency matrices of H-minor free graphs
- Boolean kernel matrices
- Shortest-path structures
- Nontrivial classes of hereditary matrices

Methodology
- Differential compression: for each $x \in \{1,\ldots,n\}$, write $\mathbf{M}_x = \mathbf{M}_y + \delta_{x,y}$, where $\delta_{x,y}$ is a "change" vector,
- Compute an approximate MST for a graph with edge weights $\|\delta_{x,y}\|_1$,
- We use computational geometry to show that matrices of VC-dimension d have MSTs of weight $O(n^{2-1/d})$,
- The algorithm stores sparse representations of $\delta_{x,y}$ and uses the in-order traversal of the MST to compute $\mathbf{M}v$.

Take a picture to download the full paper

Caution!! Approximating the VC-dimension of a set-system is Σ_3^P-hard (Mossel & Umans, 2002), so any tractable matrix-vector multiplication algorithm that exploits the VC-dimension must do so **without** knowing, computing, or approximating it!

Main Results

Thm 1 (Static $\mathbf{M}v$). *If an $n\times n$ matrix $\mathbf{M}$ has corrupted VC-dimension d: after an $\tilde{O}(n^2)$ preprocessing, there is a data structure $\mathcal{D}$ that can compute $\mathbf{M}v$ for any vector $v \in \mathbb{R}^n$ in time $O(n^{2-1/d})$ time, w.h.p.*

Thm 2 (Dynamic $\mathbf{M}v$). *If an $n \times n$ matrix $\mathbf{M}$ has corrupted VC-dimension d: after an $\tilde{O}(n^2)$ preprocessing, there is a data structure $\mathcal{D}$ that can support row and column updates (insertions/deletions) to $\mathbf{M}$ in $\tilde{O}(n)$ time. Upon querying $\mathcal{D}$ with a vector $v \in \mathbb{R}^n$, it outputs $\mathbf{M}v$ in time $O(n^{2-1/d^*})$, where d^* is the largest corrupted VC-dimension of $\mathbf{M}$ throughout its update history.*

Sauer-Shelah's Lemma: If VC($\mathbf{M}$) $= d$, then VC($\mathbf{M}^\top$) $\leq 2^d$. To avoid this blow-up, the query complexity of the algorithm actually only depends on $\min\{\text{VC}(\mathbf{M}), \text{VC}(\mathbf{M}^\top)\}$.

Pollard-pseudodimension is a popular extension of the VC-dimension to non-binary thresholds.

Thm 3 (Static $\mathbf{M}v$). *If an $n\times n$ matrix $\mathbf{M}$ has Pollard pseudodimension d and the number of thresholds is A: after an $\tilde{O}(An^2)$ preprocessing, there is a data structure that upon receiving a vector $v \in \mathbb{R}^n$, returns $\mathbf{M}v$ in time $O(Amn^{1-1/d})$, with high probability.*

Applications

The following applications have $\Omega(n^2)$-time lower bounds, conditional on the OMv conjecture. For structured graphs, this lower bound *can* be beaten.

(1) High-accuracy dynamic Laplacian solver. Given a graph G with corrupt VC-dimension d, there is a dynamic algorithm that maintains a Laplacian system solver: it supports queries that receive a vector $b \in \mathbb{R}^n$ and $\epsilon > 0$, and returns an ϵ-approx soln to $\mathbf{L}x^* = b$ in $\tilde{O}(n^{2-1/d}\log\frac{1}{\epsilon})$ time, where $\mathbf{L}$ is the graph Laplacian.

(2) Dynamic Effective Resistance. Given a graph G with corrupt VC-dimension d, there is a dynamic algorithm that maintains effective resistances in G: it supports queries that receive $u, v \in V$ and $\epsilon > 0$ and returns a $(1 \pm \epsilon)$-approximation of effective resistance in $\tilde{O}(n^{2-1/d}\log\frac{1}{\epsilon})$ time. Node updates take $\tilde{O}(n)$ time.

(3) Dynamic Triangle Detection Given a graph G with corrupt VC-dimension d, there is a dynamic algorithm that maintains whether G has a triangle. Node updates take $O(n^{2-1/d})$ time.

(4) Dynamic SSSP. If a dynamic unweighted undirected graph G has corrupt VC-dimension d, there is a dynamic algorithm that maintains $(1+\epsilon)$-approximate single source distances on G. Node updates take $\tilde{O}(kn^{2-1/2d}/\epsilon)$ time and querying the distance for any source node takes $O(n^{2-1/2d}/\epsilon)$ time.

(5) Dynamic Approx k-center. If G is an unweighted undirected graph with corrupt VC-dimension d, there is a dynamic algorithm for $(2+\epsilon)$-approx k-center with node update time $O(kn^{2-1/2d}/\epsilon)$.