The Structural Complexity of Matrix-Vector Multiplication

Emile Timothy Anand

Joint work with Jan van den Brand and Rose McCarty

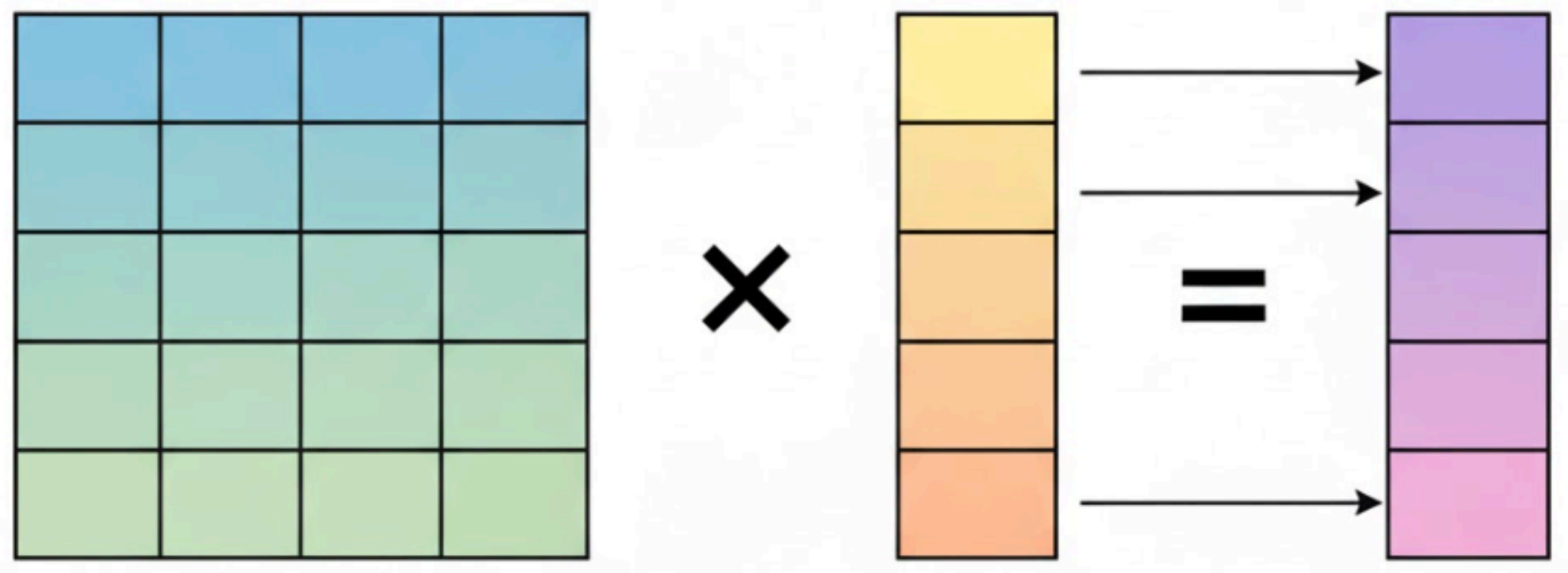
Georgia Institute of Technology

NeurIPS 2025

Problem

Preprocess a given $n \times n$ matrix $\mathbf{M}$

Support queries, that for vector $v \in \mathbb{R}^n$ return the product $\mathbf{M}v$

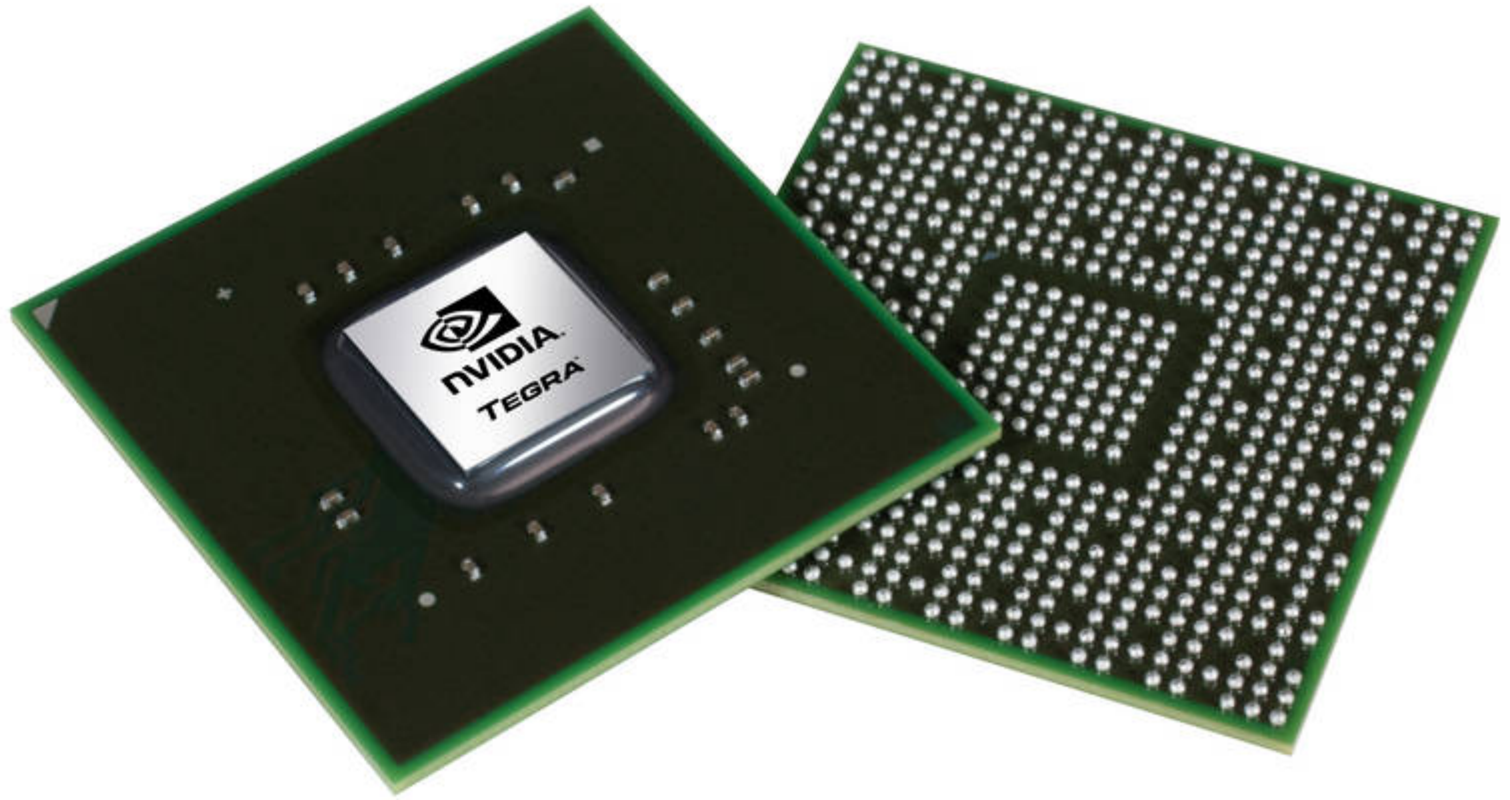


Q: Can we do queries in faster than $O(n^2)$ time?

Why does this matter?

Workhorse of iterative algorithms in machine learning

Used in optimization, computational geometry, dynamic algorithms



The current learning revolution is powered by hardware that does fast matrix-vector products

Any complexity improvement has wide-ranging implications

Prior Work

In Practice

- Sparsity-based heuristics that run in O($nnz(\mathbf{M})$) time
- In practice, they run even faster! (AMB+24, BL01, GL03)

Prior Work

In Practice

- Sparsity-based heuristics that run in O($nnz(\mathbf{M})$) time
- In practice, they run even faster!   (AMB+24, BL01, GL03)

In Theory

- $\Omega(n^2/\log n)$ worst-case time for "generic" algorithms [Clifford, Grønlund, Larsen 15]
- This also holds for the average case [Henzinger, Lincoln, Saha 22]

Prior Work

In Practice

- Sparsity-based heuristics that run in O($nnz(\mathbf{M})$) time
- In practice, they run even faster! (AMB+24, BL01, GL03)

In Theory

- $\Omega(n^2/\log n)$ worst-case time for "generic" algorithms [Clifford, Grønlund, Larsen 15]
- This also holds for the average case [Henzinger, Lincoln, Saha 22]

The Online Matrix-Vector (OMv) Conjecture

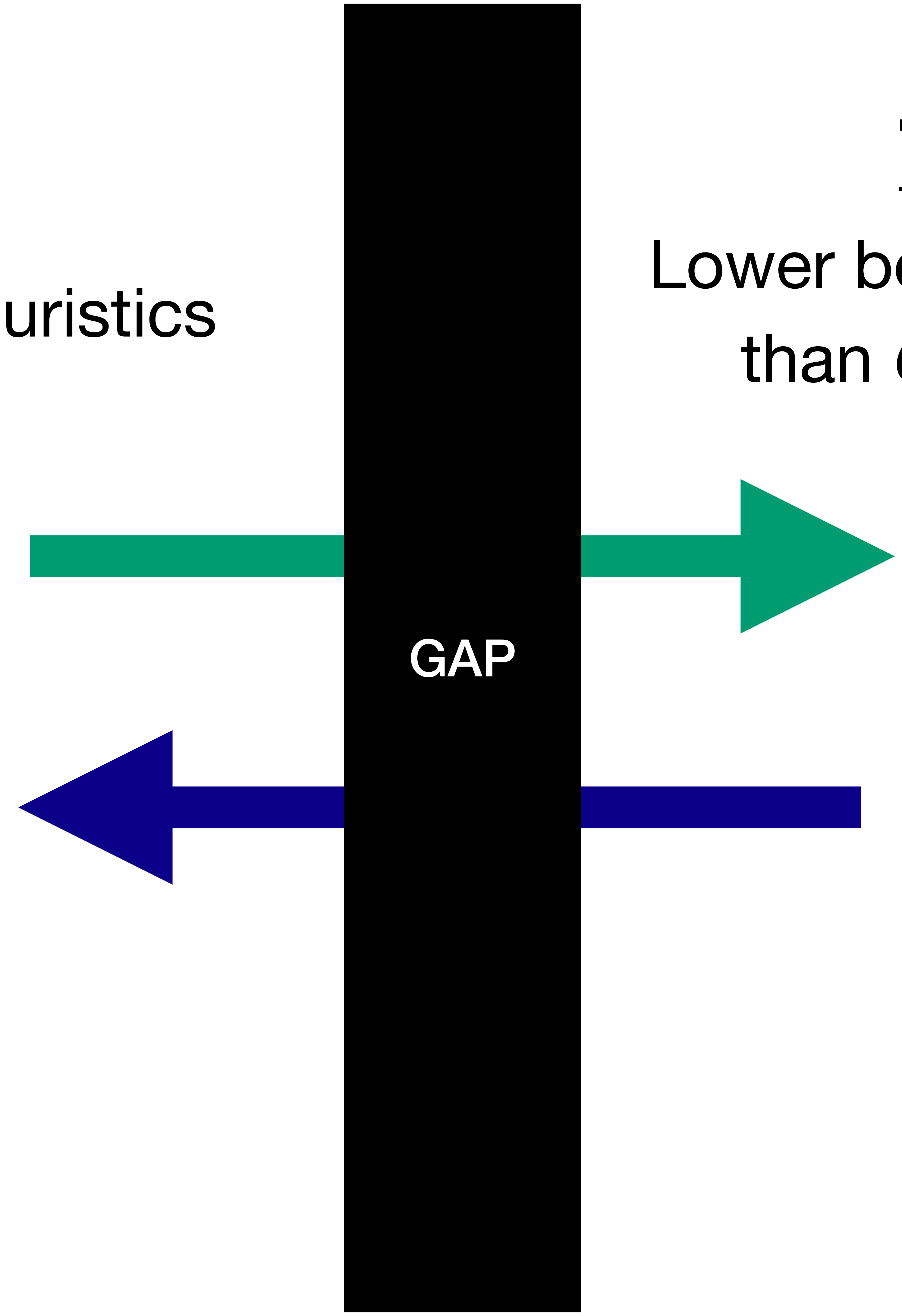
- Even for Boolean inputs, there is no algorithm that runs in $O(n^{2-\epsilon})$ worst-case time (with $\mathrm{poly}(n)$-time preprocessing)
- Open theoretical problem for > 10 years! [HKNS15]

Prior Work

Practice:

Highly efficient heuristics

Empirically faster algorithms



Theory:

Lower bound: Can't be solved faster than $O(n^2)$ time in worst-case

Provable guarantees, but **slow** algorithms.

Prior Work

Practice:

Highly efficient heuristics

Empirically faster algorithms

GAP

Theory:

Lower bound: Can't be solved faster than $O(n^2)$ time in worst-case

Provable guarantees, but **slow** algorithms.

Q) Can we bridge the gap between theory and practice?

Prior Work

Practice:

Highly efficient heuristics

Empirically faster algorithms

GAP

Theory:

Lower bound: Can't be solved faster than $O(n^2)$ time in worst-case

Provable guarantees, but **slow** algorithms.

Q) Can we bridge the gap between theory and practice?

Q) Can we use this to create even faster matrix-vector algorithms?

Structured Matrices

For certain matrices… $\mathbf{M}v$ can be done faster!

- Sparse matrices

- If the matrix is Vandermonde, Toeplitz, Hankel, or Cauchy
    - convolutional transformation algorithms $\implies$ $O(n \log n)$ multiplication

Structured Matrices

For certain matrices… $\mathbf{M}v$ can be done faster!

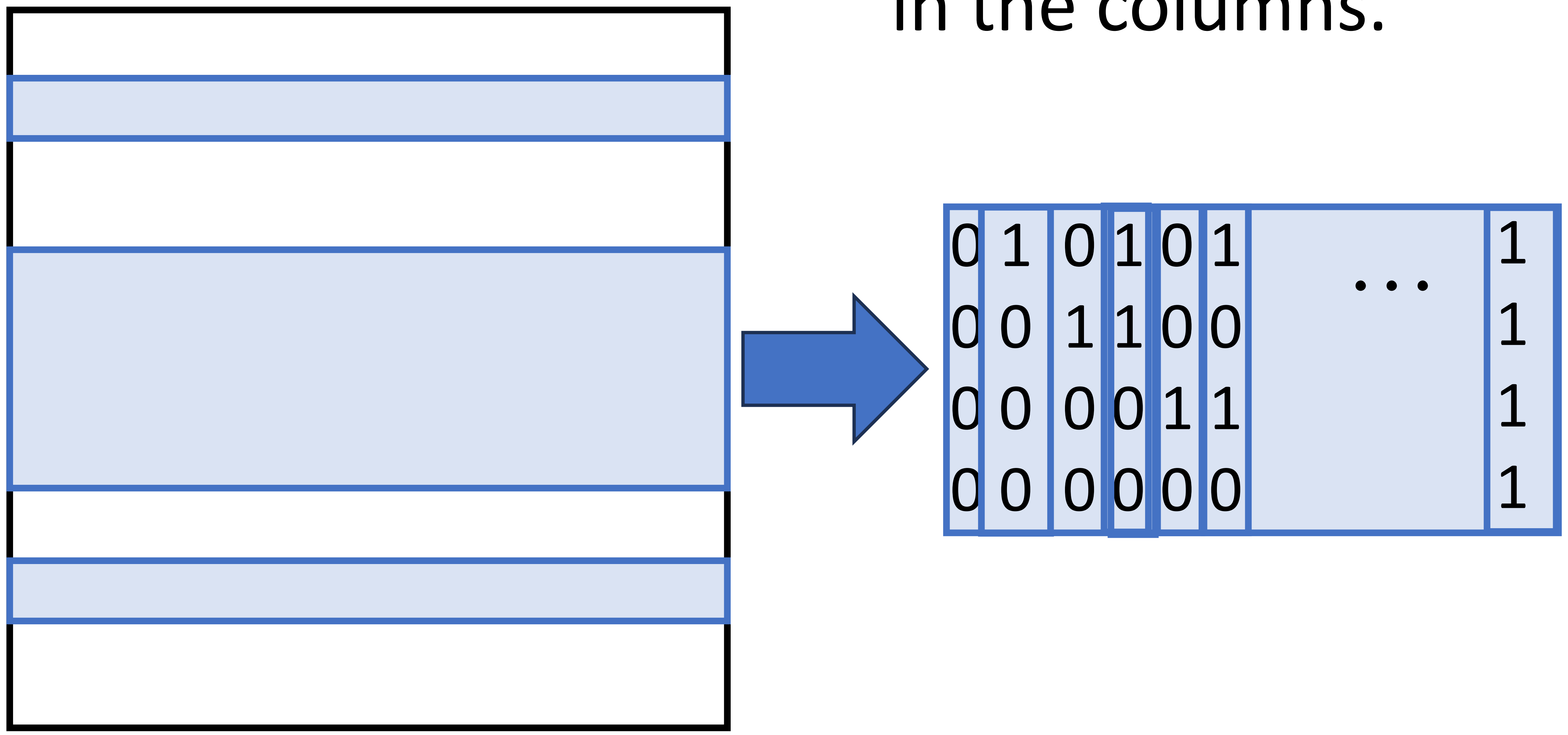
- Sparse matrices
- If the matrix is Vandermonde, Toeplitz, Hankel, or Cauchy
    - convolutional transformation algorithms $\implies$ $O(n \log n)$ multiplication

These previous results hold only for *very specific matrices*…

We need a broader parameterization of structural complexity!

VC-dimension (Vapnik–Chervonenkis)

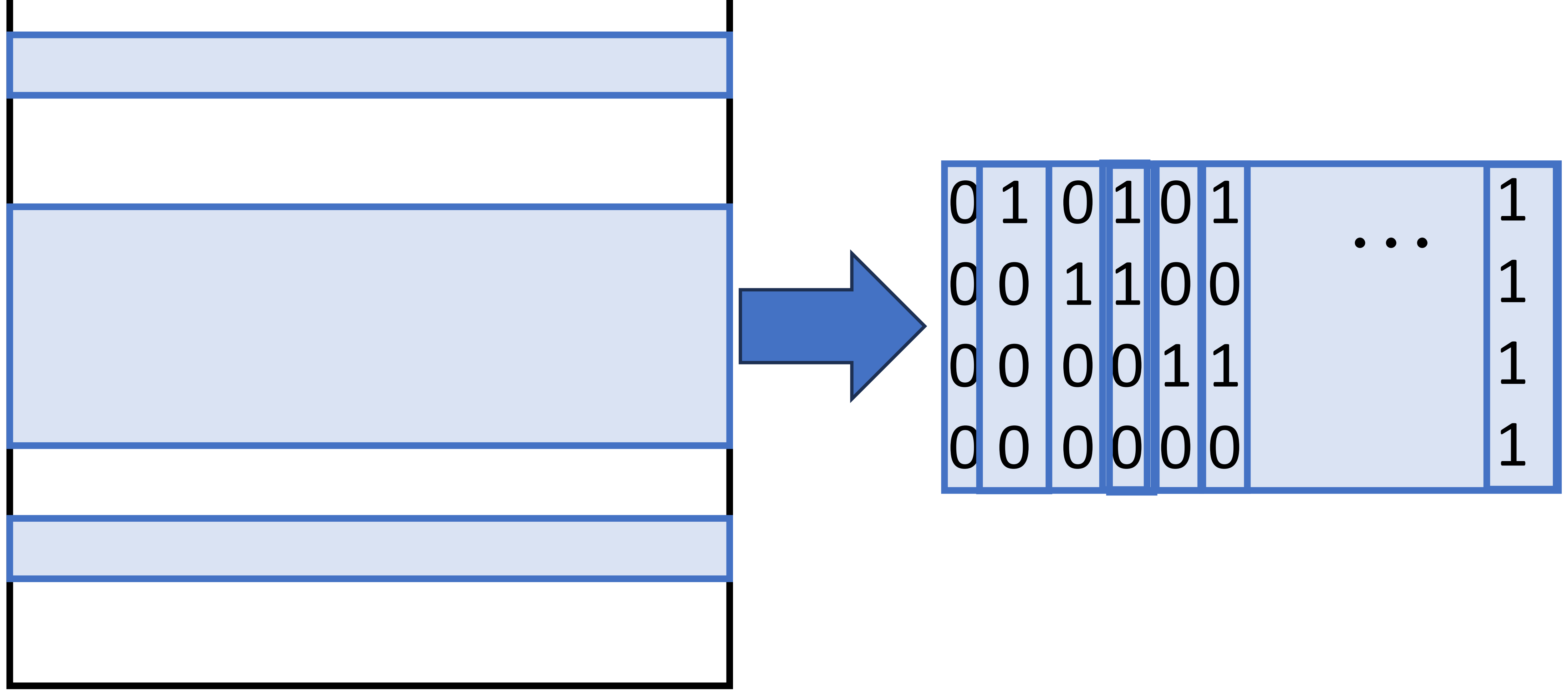
Size of largest set of rows, containing all 0/1-strings
in the columns.



- VC-dimension of real-world graphs: $3-8$.
[Coudert, Csikós, Ducoffe, Viennot’24]

VC-dimension (Vapnik–Chervonenkis)

Size of largest set of rows, containing all 0/1-strings in the columns.



Has to exclude some matrices

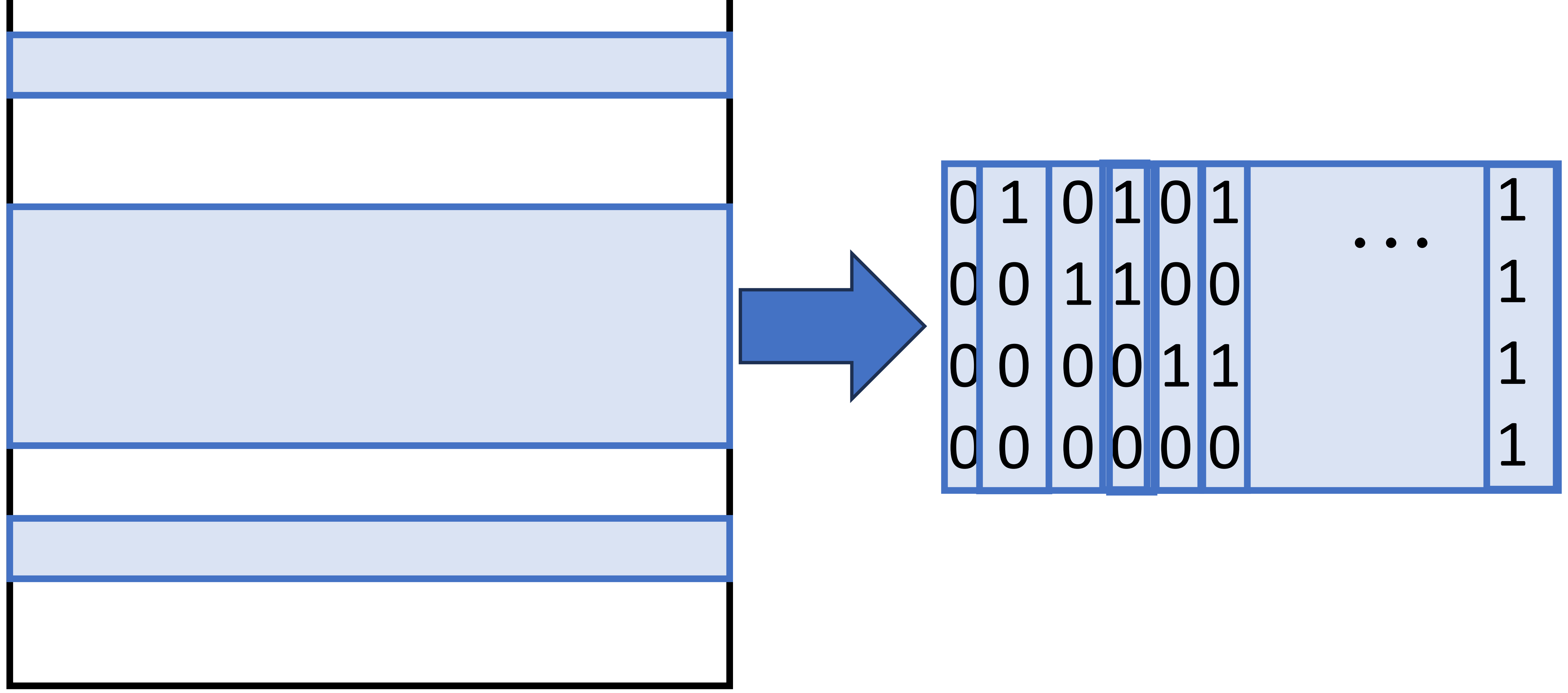
Closed under row/column deletion

Fact: Any non-trivial hereditary family P of 0/1 matrices has constant VC-dimension.

- VC-dimension of real-world graphs: $3-8$.
  [Coudert, Csikós, Ducoffe, Viennot’24]

VC-dimension (Vapnik–Chervonenkis)

Size of largest set of rows, containing all 0/1-strings in the columns.



Has to exclude some matrices

Closed under row/column deletion

Fact: Any non-trivial hereditary family P of 0/1 matrices has constant VC-dimension.

We show to solve OMv on P in $O(n^{2-\epsilon})$ time!

Our algorithm does not need to know P or the VC-dimension

- VC-dimension of real-world graphs: $3 - 8$. [Coudert, Csikós, Ducoffe, Viennot'24]

What do we show?

- For matrices with VC-dimension d, the matrix-vector multiplication problem can be solved with $\tilde{O}(n^2)$ preprocessing and $\tilde{O}(n^{2-1/d})$ query time,

What do we show?

Can handle dynamic updates to the matrix in $\tilde{O}(n)$ time

- For matrices with VC-dimension d, the matrix-vector multiplication problem can be solved with $\tilde{O}(n^2)$ preprocessing and $\tilde{O}(n^{2-1/d})$ query time,

What do we show?

Can handle dynamic updates to the matrix in $\tilde{O}(n)$ time

- For matrices with VC-dimension d, the matrix-vector multiplication problem can be solved with $\tilde{O}(n^2)$ preprocessing and $\tilde{O}(n^{2-1/d})$ query time,

- Since real-world matrices have low VC-dimensions, our result explains why the problem can be solved so much faster in practice,

What do we show?

Can handle dynamic updates to the matrix in $\tilde{O}(n)$ time

- For matrices with VC-dimension d, the matrix-vector multiplication problem can be solved with $\tilde{O}(n^2)$ preprocessing and $\tilde{O}(n^{2-1/d})$ query time,
- Since real-world matrices have low VC-dimensions, our result explains why the problem can be solved so much faster in practice,
- Our result extends to rectangular matrices, and the class of non-Boolean matrices with the analogous Pollard pseudo dimension measure,

What do we show?

Can handle dynamic updates to the matrix in $\tilde{O}(n)$ time

- For matrices with VC-dimension d, the matrix-vector multiplication problem can be solved with $\tilde{O}(n^2)$ preprocessing and $\tilde{O}(n^{2-1/d})$ query time,
- Since real-world matrices have low VC-dimensions, our result explains why the problem can be solved so much faster in practice,
- Our result extends to rectangular matrices, and the class of non-Boolean matrices with the analogous Pollard pseudo dimension measure,
- We also give an algorithm to handle the case where $\mathbf{M}$ may have VC-dimension 2^d but $\mathbf{M}^\top$ has lower VC-dimension in $\tilde{O}(n^{2-1/d})$ query time

What do we show?

Can handle dynamic updates to the matrix in $\tilde{O}(n)$ time

- For matrices with VC-dimension d, the matrix-vector multiplication problem can be solved with $\tilde{O}(n^2)$ preprocessing and $\tilde{O}(n^{2-1/d})$ query time,

- Since real-world matrices have low VC-dimensions, our result explains why the problem can be solved so much faster in practice,

- Our result extends to rectangular matrices, and the class of non-Boolean matrices with the analogous Pollard pseudo dimension measure,

- We also give an algorithm to handle the case where $\mathbf{M}$ may have VC-dimension 2^d but $\mathbf{M}^\top$ has lower VC-dimension in $\tilde{O}(n^{2-1/d})$ query time

Our proofs combine methods and insights from the areas of computational geometry, structural graph theory, and dynamic algebraic algorithms!

Conclusion

- **Sub-quadratic time OMv for structured inputs.**
- **Evidence why heuristics work well in practice.**
- **New tool for dynamic algorithms.**

Thanks!