Anticoncentration Regularizers for Stochastic Combinatorial Problems

Shiva Kaul
CMU

Geoff Gordon
CMU

The Machine Learning Tragedy

dramatic structure in IV acts

I. the gods pose a learning problem

II. the protagonist finds a statistically ideal way to solve it

III. his way is cursed by NP-hardness

IV. he is banished to statistical inferiority

The sparsity recovery problem

The setting:	$h^* \in \mathbb{R}^D$ is unknown
	$\text{supp}(h^*)$ are the positions of its S nonzero entries
	S is a constant fraction of D
Given for $1 \leq m \leq M$:	$x_m \sim N(0,1)^D$
	$y_m = \langle h^*, x_m \rangle + g_m$ for some $g_m \sim N(0, \sigma^2)$
Asymptotic reliability:	$\mathbb{P}(\text{supp}(h) \neq \text{supp}(h^*)) \to 0$ as $M, S, D \to \infty$
	with respect to the randomness of x_m and g_m

A direct approach is optimal… [W09]

$M = \Omega(S)$ is *necessary* for asymptotic reliability.

$M = O(S)$ is *sufficient* for asymptotic reliability of:

$$\min_h . \ \frac{1}{M}\sum_m (\langle h, x_m \rangle - y_m)^2 \quad \text{s.t.} \quad \|h\|_0 \leq S \qquad (\textsc{Direct}_2)$$

Still NP-hard when the numerical values of the inputs are polynomial in their bit-lengths.

↑

…but (generally) strongly NP-hard.

Convex relaxation is suboptimal.

[W06]: $M = \Theta(S \log(D - S))$ is necessary for asymptotic reliability of

$$\min_h . \ \frac{1}{M} \sum_m (\langle h, x_m \rangle - y_m)^2 + \lambda \|h\|_1 \qquad \text{(Lasso)}$$

Let's tweak $\textsc{Direct}_2$

Actually, a linear variant inheriting strong NP-hardness. (It's easier to write down.)

$$\min_h . \ \frac{1}{M}\sum_m |\langle h, x_m\rangle - y_m| \ \text{ s.t. } \ \|h\|_0 \leq S \qquad (\textsc{Direct}_1)$$

Assumption for talk: $\textsc{Direct}_1 \cong \textsc{Direct}_2$

↑

Asymptotically reliable at same rate, up to constants.

Theorem: a randomized polynomial-time algorithm is asymptotically reliable given $M = O(S)$.

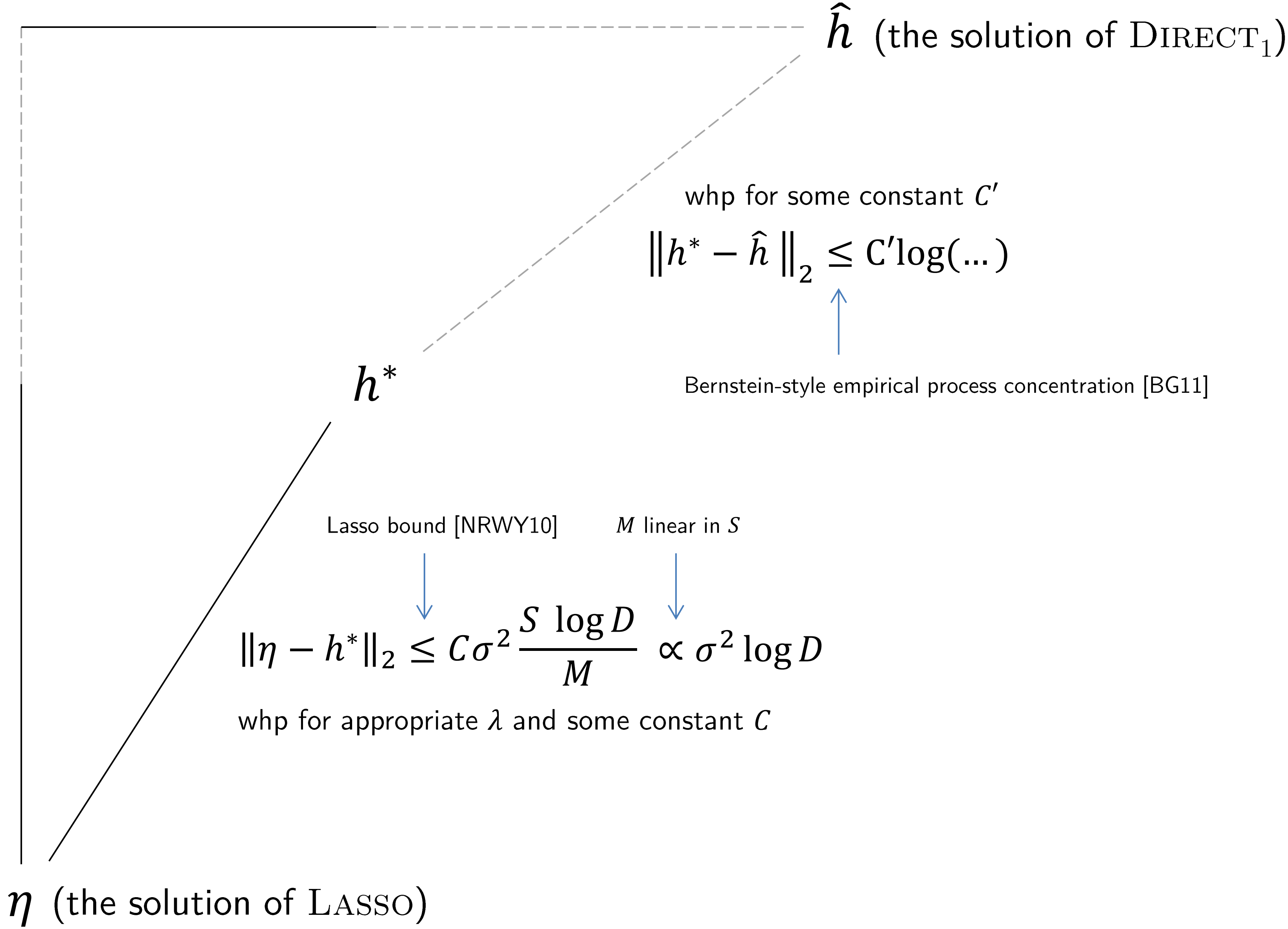
1. $\textsc{Direct}_1 \cong \textsc{Round}$, where the decision variables are rounded to take polynomially many values..

   use $\textsc{Lasso}$ *as a starting point to seed the algorithm*

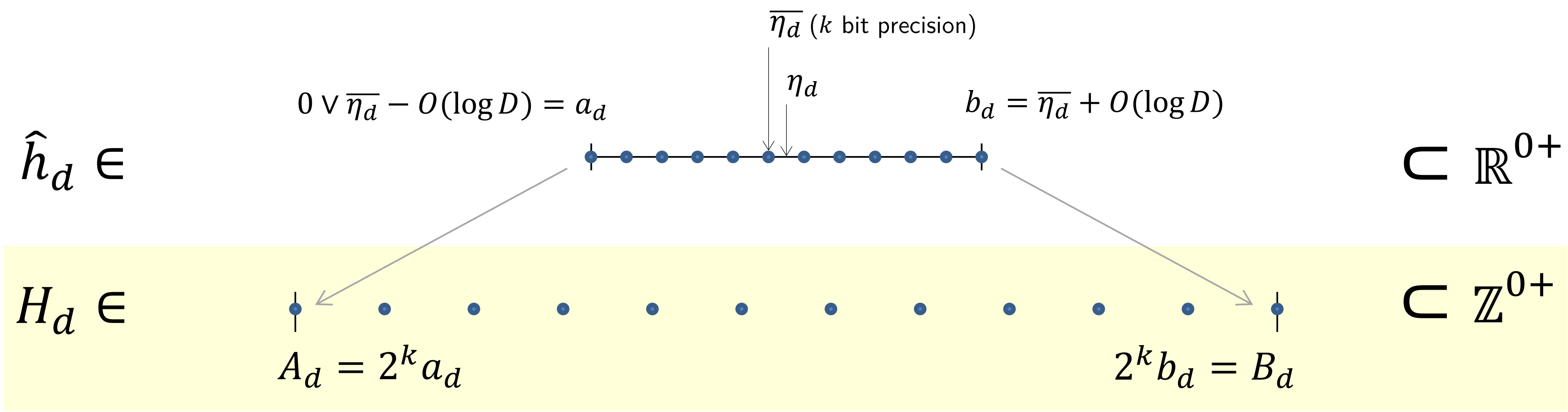
$\textsc{Direct}_1$

$$
\begin{aligned}
\text{min.}\quad & \textstyle\sum_m \ell_m / M \\
\text{s.t.}\quad & \forall m \in \{1, \dots, M\},\ d \in \{1, \dots, 2D\},\ d' \in \{1, \dots, D\} \\
& \ell_m = p_m + n_m \\
& \langle h, x_m \rangle - y_m = p_m - n_m \\
& h_d \le I_d \cdot ? \\
& I_{d'} + I_{d'+D} \le 1 \\
& \textstyle\sum_d I_d \le S \\
& \ell_m, p_m, n_m, h_d \ge 0 \\
& I_d \in \{0,1\}
\end{aligned}
$$

The real-valued decision variables can take uncountably many values.



$\hat{h}$ (the solution of DIRECT$_1$)
whp for some constant C'
$\|h^* - \hat{h}\|_2 \leq C' \log(\dots)$
Bernstein-style empirical process concentration [BG11]
h^*
Lasso bound [NRWY10]
M linear in S
$\|\eta - h^*\|_2 \leq C\sigma^2 \frac{S \log D}{M} \propto \sigma^2 \log D$
whp for appropriate λ and some constant C
η (the solution of LASSO)



We can choose k so that the number of points is polynomial, yet $\left\| \hat{h} - H/2^k \right\|_2$ is exponentially small.

ROUND (in progress)

min. $\sum_m \ell_m / M$

s.t. $\forall m \in \{1,\dots,M\},\ d \in \{1,\dots,2D\},\ d' \in \{1,\dots,D\}$

$$\ell_m = p_m + n_m$$

$$\langle H/2^k, x_m\rangle - y_m = p_m - n_m$$

$$H_d \leq I_d B_d$$

$$I_{d'} + I_{d'+D} \leq 1$$

$$\sum_d I_d \leq S$$

$$\ell_m, p_m, n_m \geq 0$$

$$H_d \in \{A_d, \dots, B_d\}$$

$$I_d \in \{0,1\}$$

Round (almost)

$$\text{min.} \quad \textstyle\sum_m L_m / M$$

$$\text{s.t.} \quad \forall m \in \{1, \dots, M\},\ d \in \{1, \dots, 2D\},\ d' \in \{1, \dots, D\}$$

$$L_m = P_m + N_m$$

$$\langle H, x_m 2^k \rangle - y_m 2^k \approx P_m - N_m$$

Actually need to replace with inequalities and introduce more variables. Won't bother, since it will simplify anyway.

$$H_d \leq I_d B_d$$

$$I_{d'} + I_{d'+D} \leq 1$$

$$\textstyle\sum_d I_d \leq S$$

$$L_m, P_m, N_m \in \{\dots\}$$

$$H_d \in \{A_d, \dots, B_d\}$$

$$I_d \in \{0,1\}$$

ROUND (almost)

$$\text{min.} \quad \textstyle\sum_m L_m / M$$

$$\text{s.t.} \quad \forall m \in \{1, \dots, M\},\ d \in \{1, \dots, 2D\},\ d' \in \{1, \dots, D\}$$

$$L_m = P_m + N_m$$

$$\langle H, x_m 2^k \rangle - y_m 2^k \approx P_m - N_m$$

$$H_d \leq I_d B_d$$

$$I_{d'} + I_{d'+D} \leq 1$$

$$\textstyle\sum_d I_d \leq S$$

$$L_m, P_m, N_m \in \{\dots\}$$

$$H_d \in \{A_d, \dots, B_d\}$$

$$I_d \in \{0,1\}$$

1. $\textsc{Direct}_1 \subsetneqq \textsc{Round}$, where the decision variables are rounded to take polynomially many values.
   use LASSO *as a starting point to seed the algorithm*

2. $\textsc{Round} \subsetneqq \textsc{SmoothRound}$, which is smoothed by a random perturbation.
   may perturb a random program

SmoothRound (almost)

$$\min.\ \sum_m L_m / M$$

s.t. $\forall m \in \{1, \dots, M\},\ d \in \{1, \dots, 2D\},\ d' \in \{1, \dots, D\}$

$$L_m = P_m + N_m$$

$$\langle H, x_m 2^k \rangle - z_m 2^k \approx P_m - N_m$$

$$H_d \le I_d B_d$$

$$I_{d'} + I_{d'+D} \le 1$$

$$\sum_d I_d \le S$$

$$L_m, P_m, N_m \in \{\dots\}$$

$$H_d \in \{A_d, B_d\}$$

$$I_d \in \{0,1\}$$

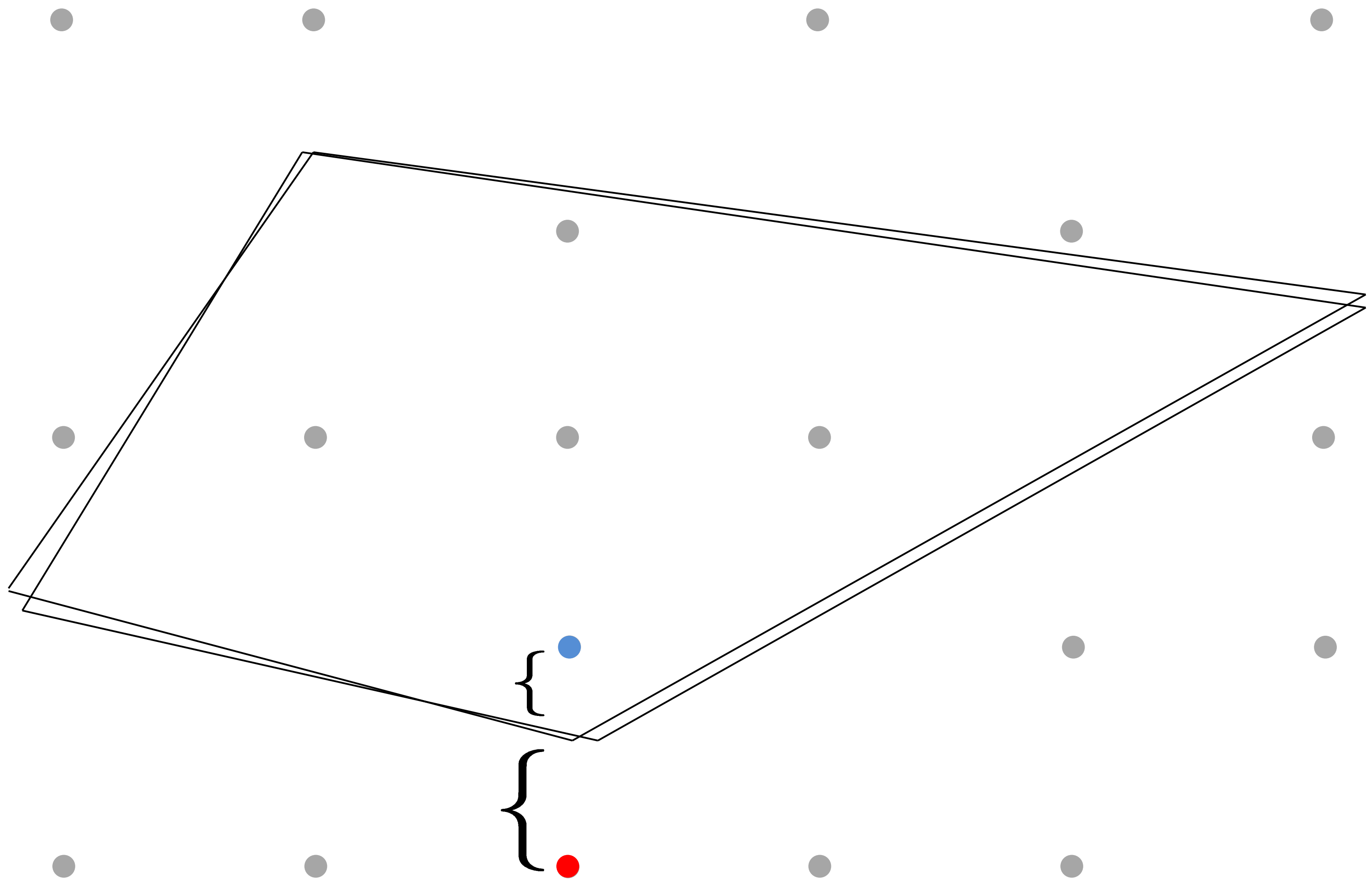
$z_m = y_m + \rho_m$ where $\rho_m \sim N(0, r^2)$. Adding it in allows us to 'restart' without drawing new sample.

1. $\textsc{Direct}_1 \cong \textsc{Round}$, where the decision variables are rounded to take polynomially many values.
   use LASSO *as a starting point to seed the algorithm*

2. $\textsc{Round} \cong \textsc{SmoothRound}$, which is smoothed by a random perturbation.
   may perturb a random program

3. SMOOTHROUND can be solved in polynomial time if ROUND can be solved in pseudopolynomial time.
   perturbed combinatorial problems have few optimal solutions

Polynomial in the numerical values of the inputs.
With this power we can solve some weakly NP-hard problems.



Smoothing leads to poly-size margins; the solution will still be ● even if the inputs (i.e. x_m and z_m) are truncated to logarithmic length i.e. polynomial value.

1. $\textsc{Direct}_1 \cong \textsc{Round}$, where the decision variables are rounded to take polynomially many values.
   use LASSO *as a starting point to seed the algorithm*

2. $\textsc{Round} \cong \textsc{SmoothRound}$, which is smoothed by a random perturbation.
   may perturb a random program

3. SMOOTHROUND can be solved in polynomial time if ROUND can be solved in pseudopolynomial time.
   perturbed combinatorial problems have few optimal solutions

4. ROUND can be solved in pseudopolynomial time.
   input can't encode complex dependencies

ROUND

$$\min.\ \sum_m L_m / M$$

s.t. $\forall m \in \{1, \dots, M\},\ d \in \{1, \dots, 2D\},\ d' \in \{1, \dots, D\}$

$$L_m = P_m + N_m$$

$$\langle H, \bar{x}_m 2^k \rangle - \bar{y}_m 2^k = P_m - N_m$$

$$H_d + \phi_d = I_d B_d$$

$$I_{d'} + I_{d'+D} + \psi_{d'} = 1$$

$$\sum_d I_d + \Psi = S$$

$$L_m, P_m, N_m \in \{\dots\}$$

$$H_d \in \{A_d, \dots, B_d\}$$

$$I_d \in \{0,1\}$$

$$\phi_d, \psi_{d'}, \Psi \in \mathbb{Z}^{0+}$$

$m \in \{1, \dots, M\}$ $d' \in \{1, \dots, D\}$

Since $M = \Omega(S) = \Omega(D)$ we may assume there are as many rows as columns.

$m \in \{1, \dots, M\}$

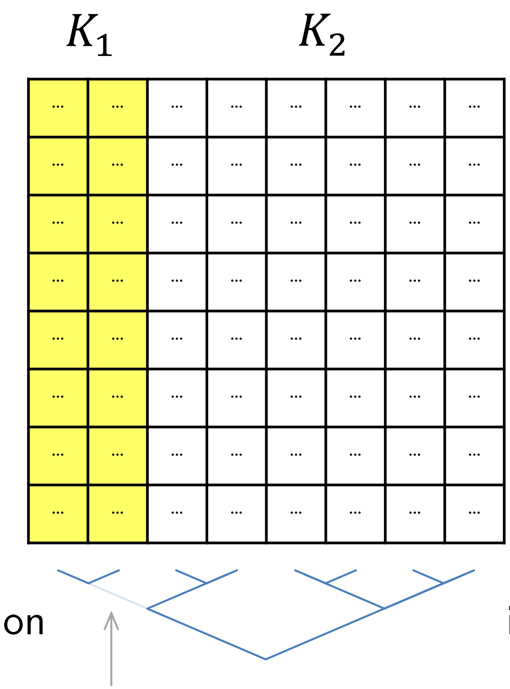
L_m	N_m	P_m	$H_{d'}$	$H_{d'+D}$	$I_{d'}$	$I_{d'+D}$	$\phi_{d'}$	$\psi_{d'}$	Ψ
1	-1	-1							
			1	1	-1		1		
					1	1		1	
	1	-1	$\bar{x}_{m,d'}$	$\bar{x}_{m,d'+D}$					
...	...	...	...	...	...	...	...	...	...

0
0
0
$\bar{y}_m$
...

It mostly encodes a matrix of iid N(0,1) random variables.

The inability to encode complex dependencies is captured by constant branchwidth.

$$K =$$



A branch decomposition is a binary tree on the columns

Cutting an edge partitions the columns into K_1 and K_2

$$\text{branchwidth} = \min_{\text{decompositions}} \max_{\text{cuts}} \left(\text{rank}(K_1) + \text{rank}(K_2) - \text{rank}(K) + 1\right)$$

[CG06]: An integer linear program in equational form can be solved in pseudopolynomial time if its decision variables take polynomially many values and its constraint matrix has constant branchwidth.

Done.

Conclusions

Don't let worst-case hardness scare you away from average-case problems.

In order to obtain better statistical guarantees, you can exploit:

- the huge amount of work on relaxations (don't just toss it out!),
- the instable, random nature of the optimization program,
- simple structure of the input.