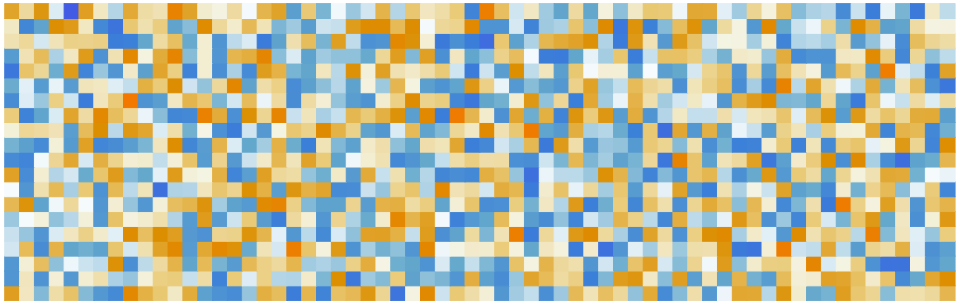
Exact Support Recovery via (Refined) Least Squares

Stefan Steinerberger

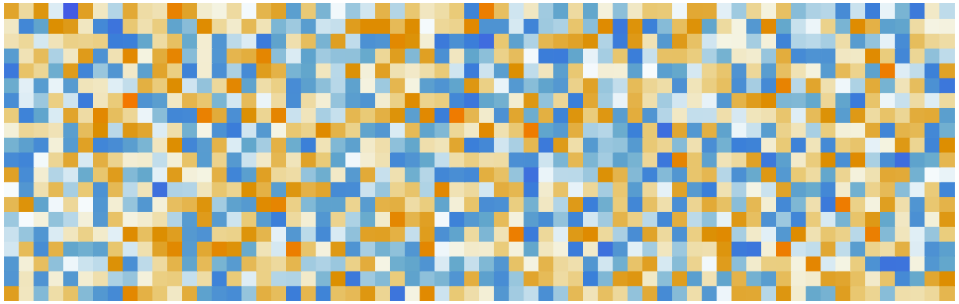
(joint with Ofir Lindenbaum)

UNIVERSITY of WASHINGTON

The Problem

$y =$  $\theta + \omega$

The Problem

$y =$  $\theta + \omega$

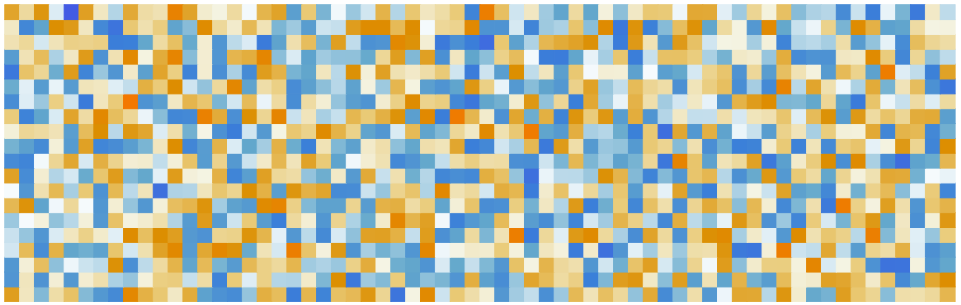
known

unknown

The Problem

Gaussian Noise

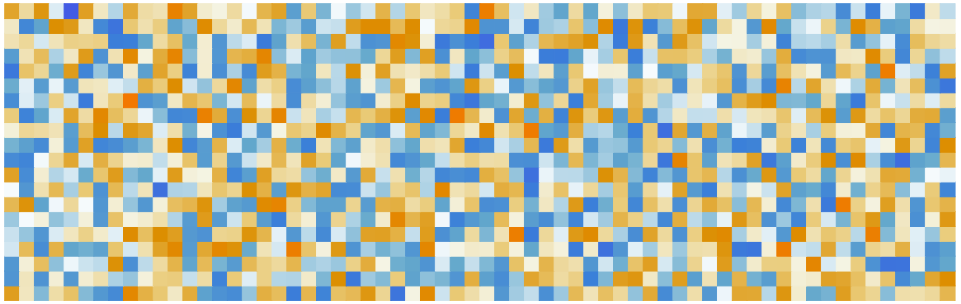
lots of 0s

$y =$  $\theta + \omega$

known

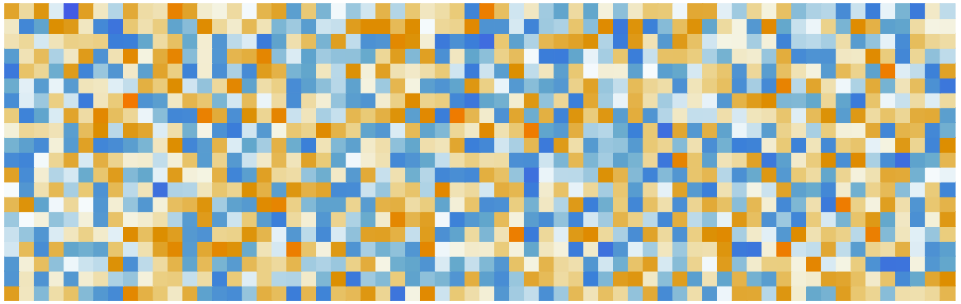
unknown

The Problem

$y =$  $\theta + \omega$

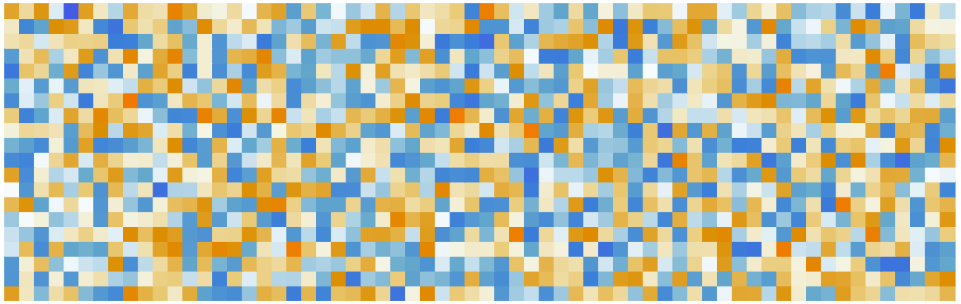
- We have $y = A\theta + \omega$. We know y and A, ω is an (unknown) random Gaussian vector. Goal: recover $\theta \in \{-1, 0, 1\}^D$.

The Problem

$y =$  $\theta + \omega$

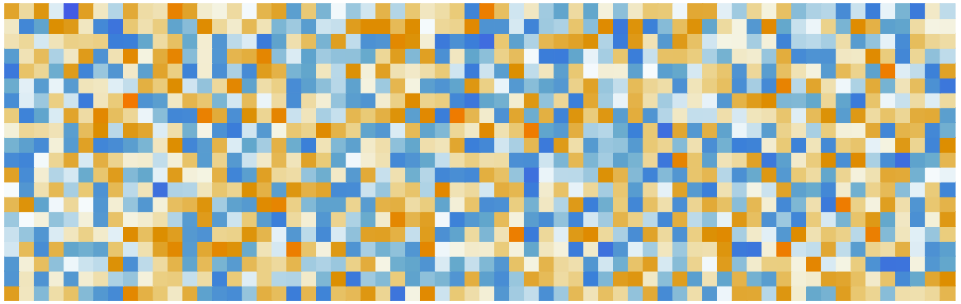
- We have $y = A\theta + \omega$. We know y and A, ω is an (unknown) random Gaussian vector. Goal: recover $\theta \in \{-1, 0, 1\}^D$.
- A common example given is gene expression: you have 30.000 genes which are turned on or off.

The Problem

$y =$  $\theta + \omega$

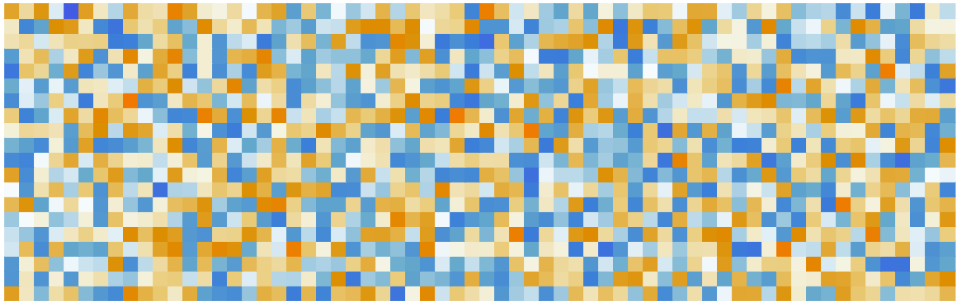
- We have $y = A\theta + \omega$. We know y and A, ω is an (unknown) random Gaussian vector. Goal: recover $\theta \in \{-1, 0, 1\}^D$.
- A common example given is gene expression: you have 30.000 genes which are turned on or off. You suspect that what is observed depends at most on 5 genes (columns of the matrix).

The Problem

$y =$  $\theta + \omega$

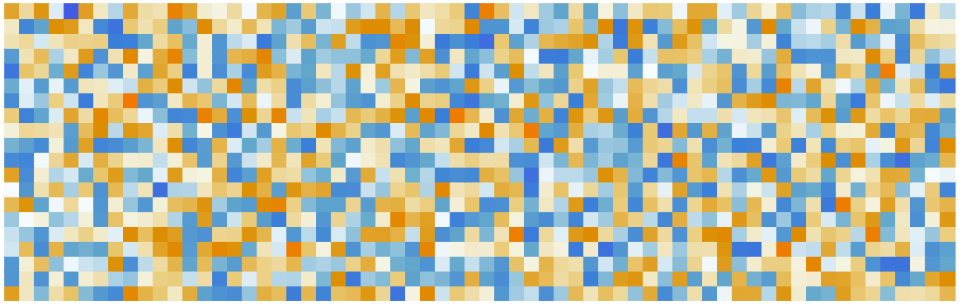
- We have $y = A\theta + \omega$. We know y and A, ω is an (unknown) random Gaussian vector. Goal: recover $\theta \in \{-1, 0, 1\}^D$.
- A common example given is gene expression: you have 30.000 genes which are turned on or off. You suspect that what is observed depends at most on 5 genes (columns of the matrix). Do you really need to run 30.000 experiments to find the 5 relevant genes?

The Problem

$y =$  $\theta + \omega$

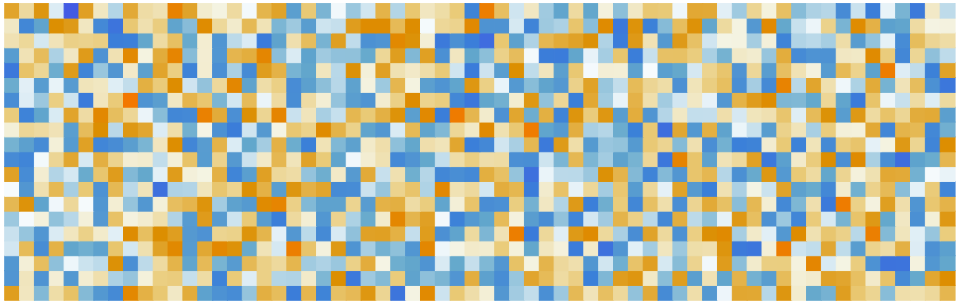
- We have $y = A\theta + \omega$. We know y and A, ω is an (unknown) random Gaussian vector. Goal: recover $\theta \in \{-1, 0, 1\}^D$.
- A common example given is gene expression: you have 30.000 genes which are turned on or off. You suspect that what is observed depends at most on 5 genes (columns of the matrix). Do you really need to run 30.000 experiments to find the 5 relevant genes? Or maybe only 50?

The Problem

$y =$  $\theta + \omega$

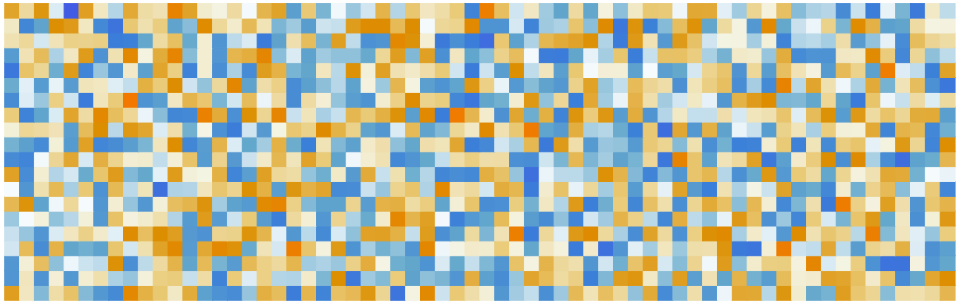
- We have $y = A\theta + \omega$. We know y and A, ω is an (unknown) random Gaussian vector. Goal: recover $\theta \in \{-1, 0, 1\}^D$.
- A common example given is gene expression: you have 30.000 genes which are turned on or off. You suspect that what is observed depends at most on 5 genes (columns of the matrix). Do you really need to run 30.000 experiments to find the 5 relevant genes? Or maybe only 50? And of course there is noise.

The Problem

$y =$  $\theta + \omega$

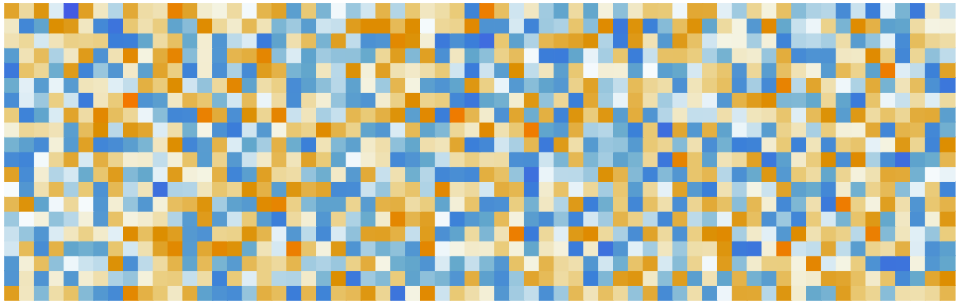
- We have $y = A\theta + \omega$. We know y and A, ω is an (unknown) random Gaussian vector. Goal: recover $\theta \in \{-1, 0, 1\}^D$.

The Problem

$y =$  $\theta + \omega$

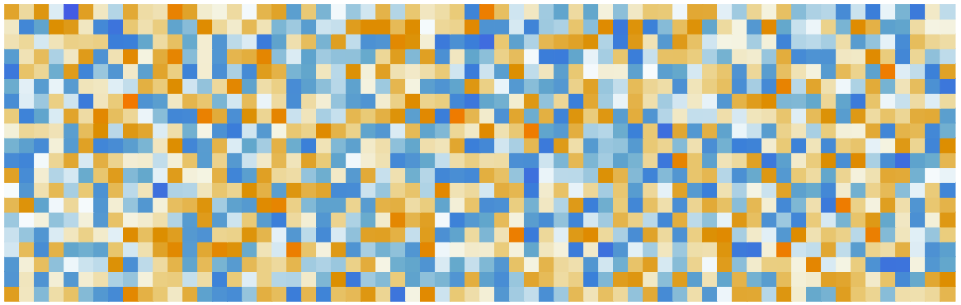
- We have $y = A\theta + \omega$. We know y and A, ω is an (unknown) random Gaussian vector. Goal: recover $\theta \in \{-1, 0, 1\}^D$.
- Clearly impossible: the system is underdetermined and we have noise. What if θ is sparse?

The Problem

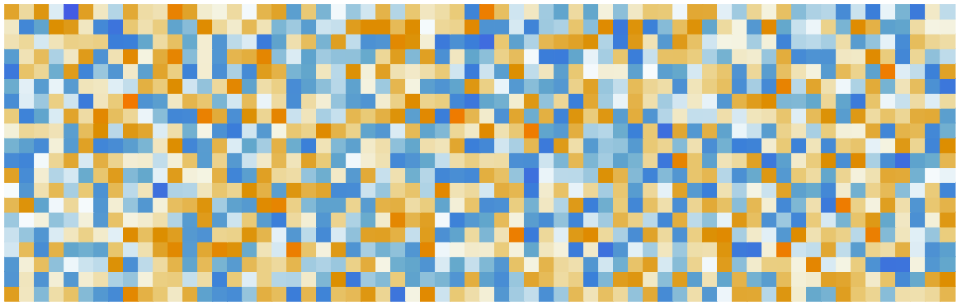
$y =$  $\theta + \omega$

- We have $y = A\theta + \omega$. We know y and A, ω is an (unknown) random Gaussian vector. Goal: recover $\theta \in \{-1, 0, 1\}^D$.
- Clearly impossible: the system is underdetermined and we have noise. What if θ is sparse?
- Formally: $A \in \mathbb{R}^{N \times D}$ and θ vanishes on all but k coordinates. I'll try to keep things variable-free as much as possible.

A first idea

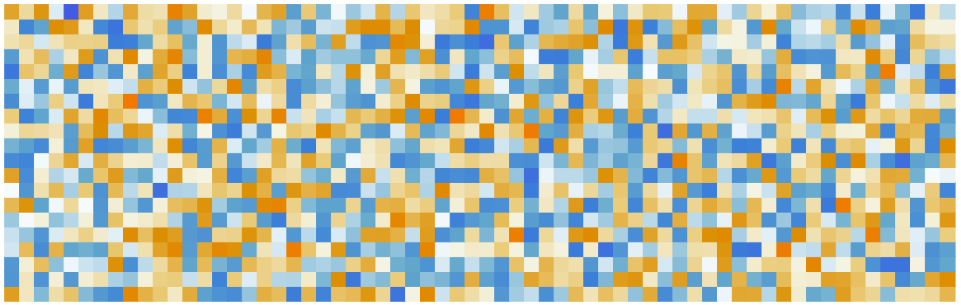
$y =$  $\theta + \omega$

A first idea

$y =$  $\theta + \omega$

Suppose that θ only has one nonzero entry in the $i-$th position.

A first idea

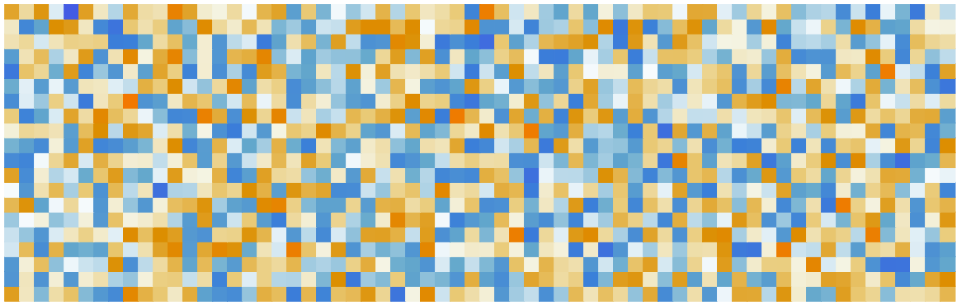
$y =$  $\theta + \omega$

Suppose that θ only has one nonzero entry in the $i-$th position. Then

$$y = \pm A_{\cdot,i} + \omega$$

and $A_{\cdot,i}$ is the $i-$th column of the matrix (but we do not know which one).

A first idea

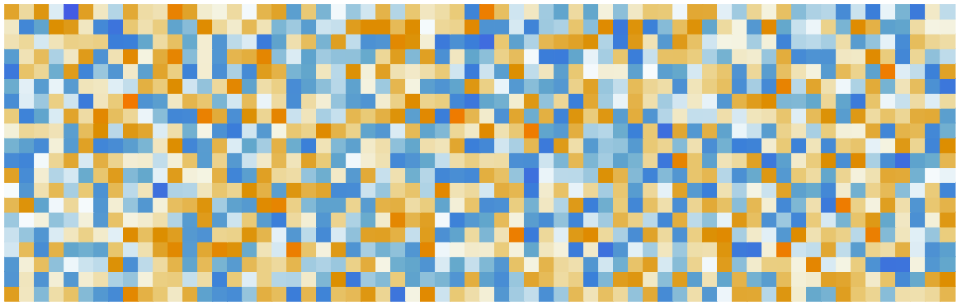
$y =$  $\theta + \omega$

Suppose that θ only has one nonzero entry in the $i-$th position. Then

$$y = \pm A_{\cdot,i} + \omega$$

and $A_{\cdot,i}$ is the $i-$th column of the matrix (but we do not know which one). This is a nice question: you have a list of vector $v_1, \ldots, v_n$ in front of you.

A first idea

$y =$  $\theta + \omega$

Suppose that θ only has one nonzero entry in the $i-$th position. Then

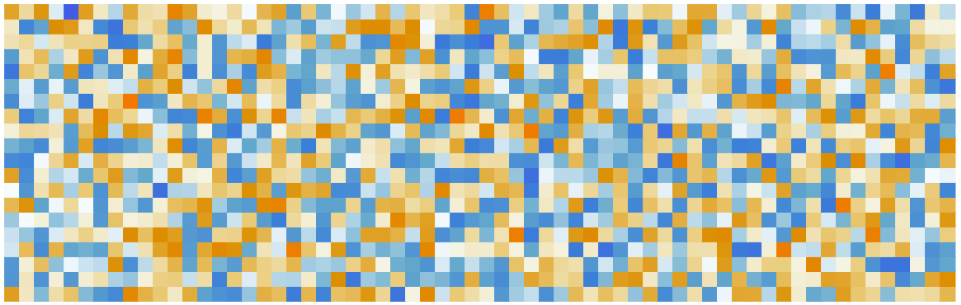
$$y = \pm A_{\cdot,i} + \omega$$

and $A_{\cdot,i}$ is the $i-$th column of the matrix (but we do not know which one). This is a nice question: you have a list of vector $v_1, \ldots, v_n$ in front of you. Somebody gives you

$$y = v_i + \omega.$$

How do you have the best chance of getting v_i?

A first idea

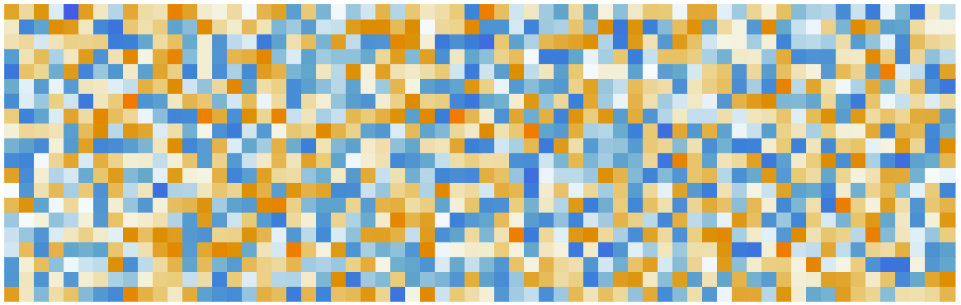
$y =$  $\theta + \omega$

Suppose that θ only has one nonzero entry in the $i-$th position. Then

$$y = \pm A_{\cdot,i} + \omega$$

and $A_{\cdot,i}$ is the $i-$th column of the matrix (but we do not know which one).

A first idea

$y =$  $\theta + \omega$

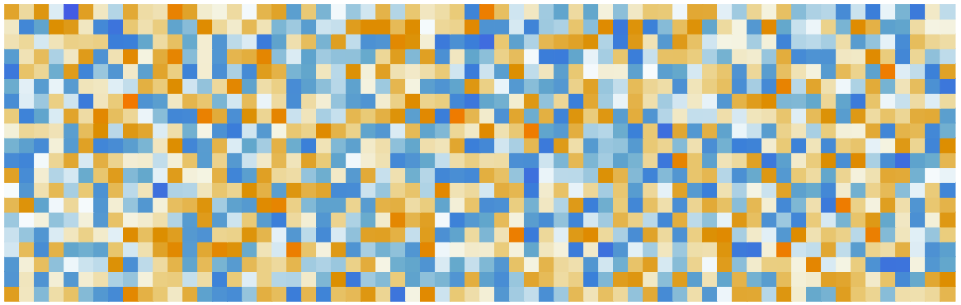
Suppose that θ only has one nonzero entry in the $i-$th position. Then

$$y = \pm A_{\cdot,i} + \omega$$

and $A_{\cdot,i}$ is the $i-$th column of the matrix (but we do not know which one). A good idea is to take the inner product

$$\langle y, A_{\cdot,j} \rangle = \langle \pm A_{\cdot,i}, A_{\cdot,j} \rangle + \langle \omega, A_{\cdot,j} \rangle$$

A first idea

$y =$  $\theta + \omega$

Suppose that θ only has one nonzero entry in the $i-$th position. Then

$$y = \pm A_{\cdot,i} + \omega$$

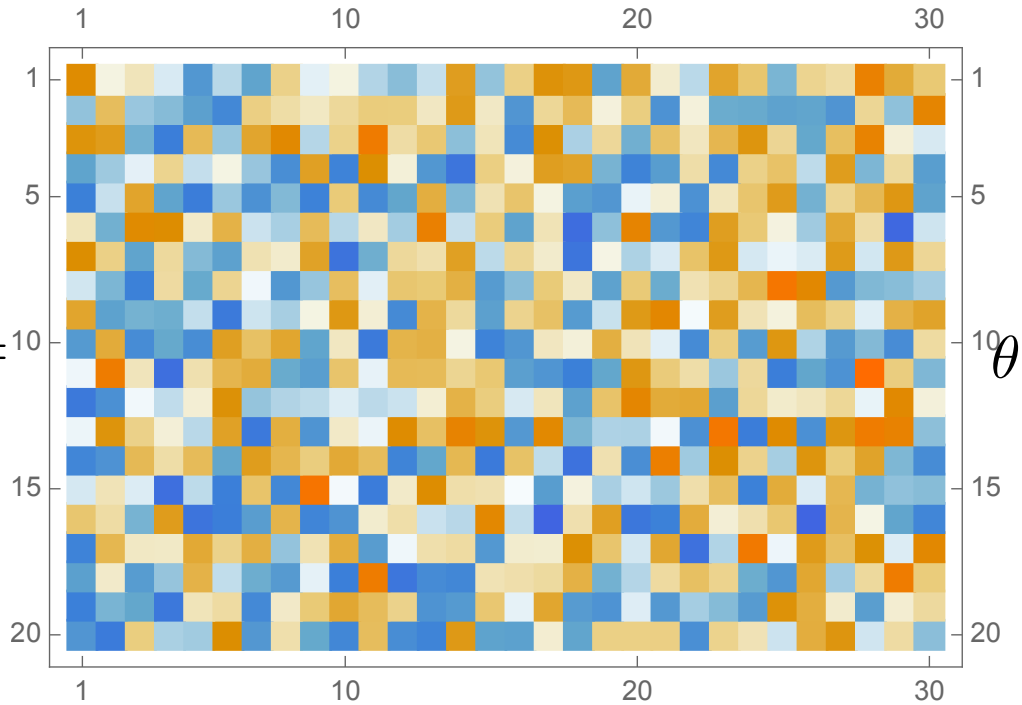
and $A_{\cdot,i}$ is the $i-$th column of the matrix (but we do not know which one). A good idea is to take the inner product

$$\langle y, A_{\cdot,j} \rangle = \langle \pm A_{\cdot,i}, A_{\cdot,j} \rangle + \langle \omega, A_{\cdot,j} \rangle$$

The first term is only big when $i = j$, the second is always equally random. Pick the j for which the inner product is the largest.

Example!

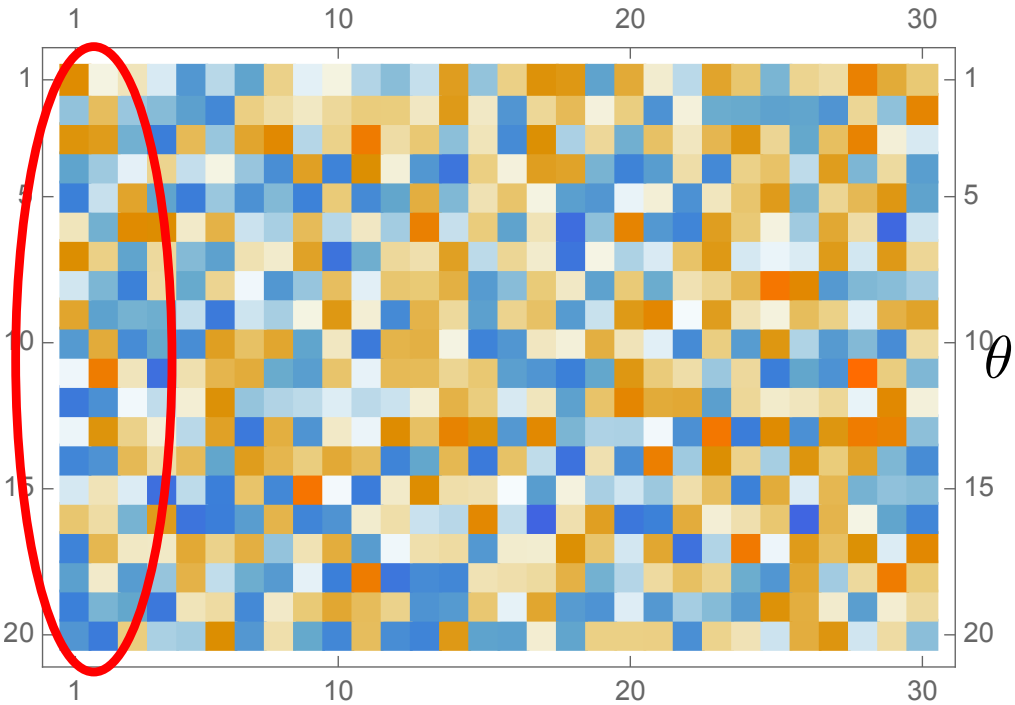
We take the following matrix $A \in \mathbb{R}^{20\times 30}$.

$y =$  $\theta + \omega$

where $\theta = (1, 1, 0, 0, \dots)$ and $\omega_i \sim \mathcal{N}(0, 1)$. How to get θ from y?

Example!

We take the following matrix $A \in \mathbb{R}^{20\times 30}$.

$y =$  $\theta + \omega$

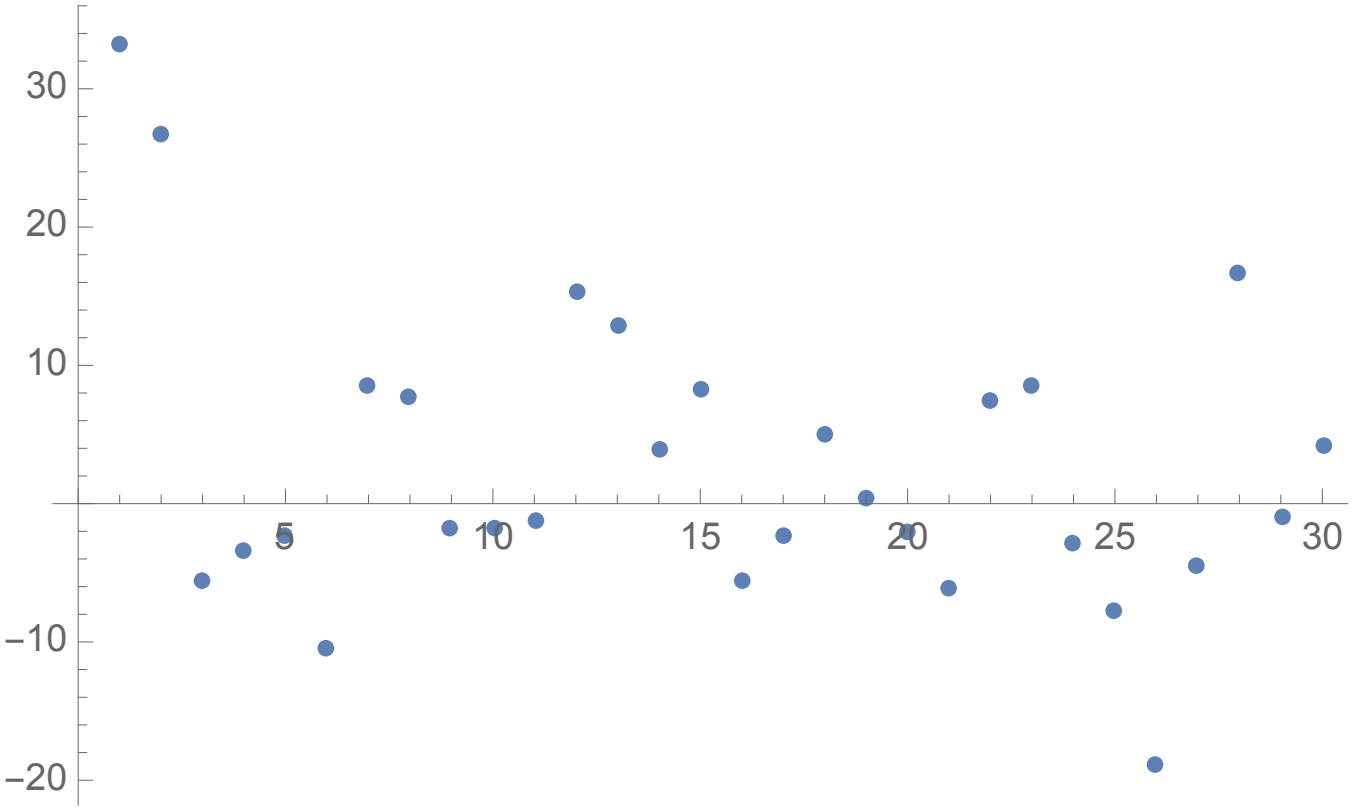
where $\theta = (1, 1, 0, 0, \dots)$ and $\omega_i \sim \mathcal{N}(0, 1)$. How to get θ from y?

Example!

Let's take inner products of the y with the columns.

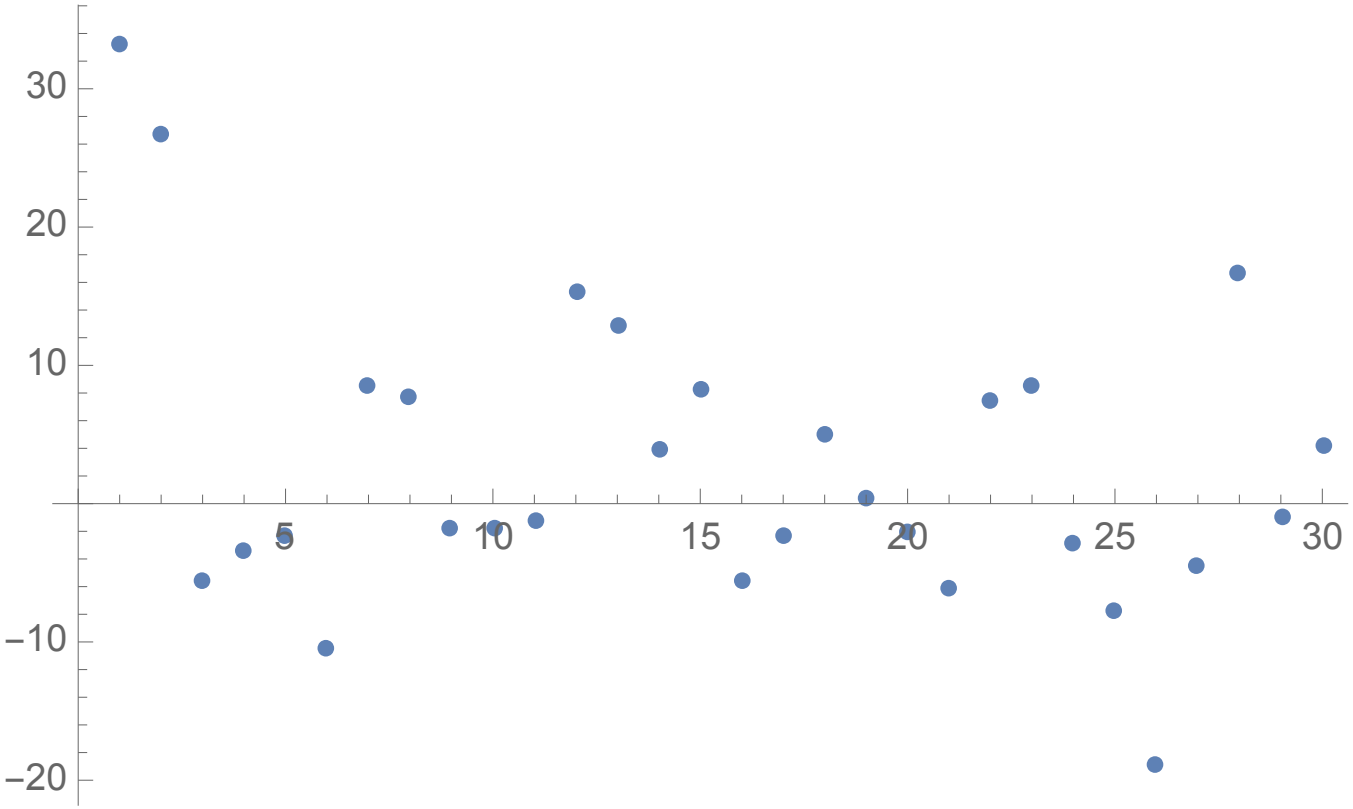
Example!

Let's take inner products of the y with the columns.

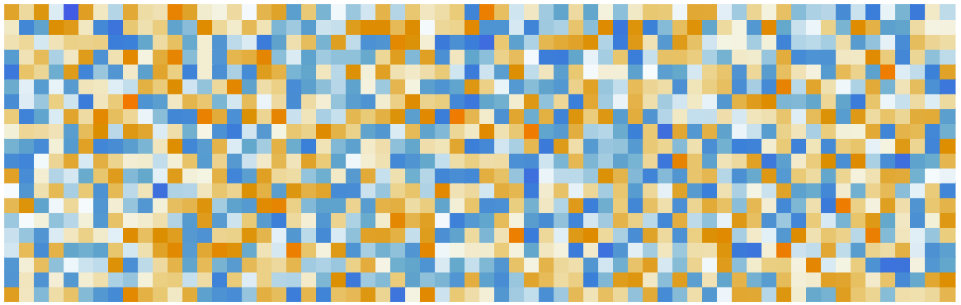


Example!

Let's take inner products of the y with the columns.



Clearly the first two coordinates stick out.

$y =$  $\theta + \omega$

Such methods are known as **Matching Pursuit** (Mallat & Zhang, Gilbert & Tropp). There are many variations on it, for example RandOMP (Elad & Yavneh), regularized OMP (Needell & Vershynin), ...

Other approach: Lasso

$$y = A\theta + w.$$

Other approach: Lasso

$$y = A\theta + w.$$

The Lasso (Tibshirani 1996)

$$\|y - Ax\|_2^2 \to \min$$
$$\|x\|_1 \leq R$$

Other approach: Lasso

$$y = A\theta + w.$$

The Lasso (Tibshirani 1996)

$$\|y - Ax\|_2^2 \to \min$$
$$\|x\|_1 \leq R$$

The ℓ^1-norm 'encourages' sparsity, a very influential idea.

Other approach: Lasso

$$y = A\theta + w.$$

The Lasso (Tibshirani 1996)

$$\|y - Ax\|_2^2 \to \min$$
$$\|x\|_1 \leq R$$

The ℓ^1-norm 'encourages' sparsity, a very influential idea. Many variations, such as

$$\|y - Ax\|_2^2 + \lambda \cdot \|x\|_1^2 \to \min\,.$$

Other approach: Lasso

$$y = A\theta + w.$$

The Lasso (Tibshirani 1996)

$$\|y - Ax\|_2^2 \to \min$$
$$\|x\|_1 \leq R$$

The ℓ^1-norm 'encourages' sparsity, a very influential idea. Many variations, such as

$$\|y - Ax\|_2^2 + \lambda \cdot \|x\|_1^2 \to \min .$$

One version we will also consider is the 'Trimmed Lasso' where sparsity is enforced by

$$T_k(x) = \min_{\|\phi\|_0 \leq k} \|x - \phi\|_1.$$

Other approaches

Because of time constraints, I am not going to explain

- Iterative Support Detection (ISD), Wang & Yin 2010
- Iterated Reweighted ℓ^1-minimization (IRL1), Candes, Wakin & Boyd (2008)

Other approaches

Because of time constraints, I am not going to explain

- Iterative Support Detection (ISD), Wang & Yin 2010
- Iterated Reweighted ℓ^1-minimization (IRL1), Candes, Wakin & Boyd (2008)

There are also many other methods.

How do these methods compare?

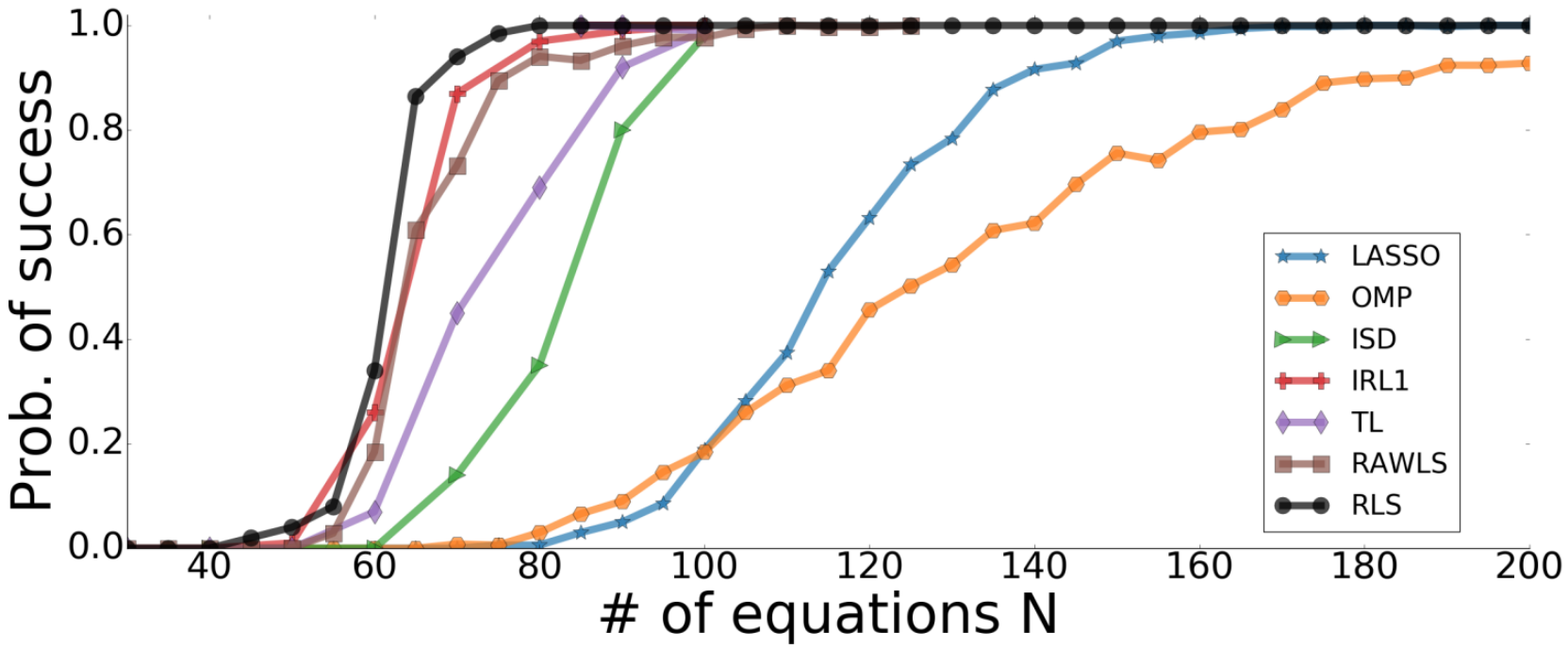
64 unknown variables (say, genes) and $\theta \in \{-1, 0, 1\}^{64}$ has 30 nonzero entries (not that sparse) and $\mathcal{N}(0, 1)$ Gaussian noise.

How do these methods compare?

64 unknown variables (say, genes) and $\theta \in \{-1, 0, 1\}^{64}$ has 30 nonzero entries (not that sparse) and $\mathcal{N}(0, 1)$ Gaussian noise. How many 'experiments' (equations) do you need to recover θ?

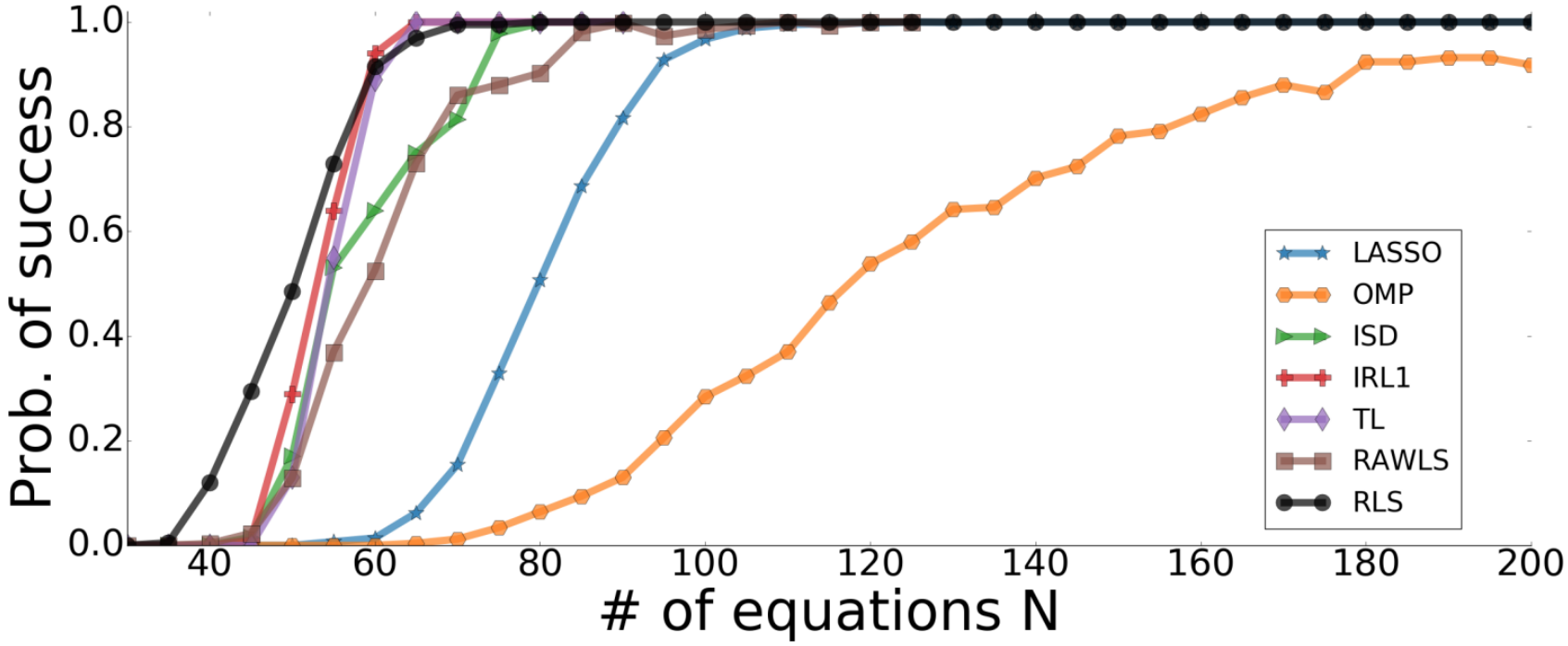
How do these methods compare?

64 unknown variables (say, genes) and $\theta \in \{-1, 0, 1\}^{64}$ has 30 nonzero entries (not that sparse) and $\mathcal{N}(0, 1)$ Gaussian noise. How many 'experiments' (equations) do you need to recover θ?



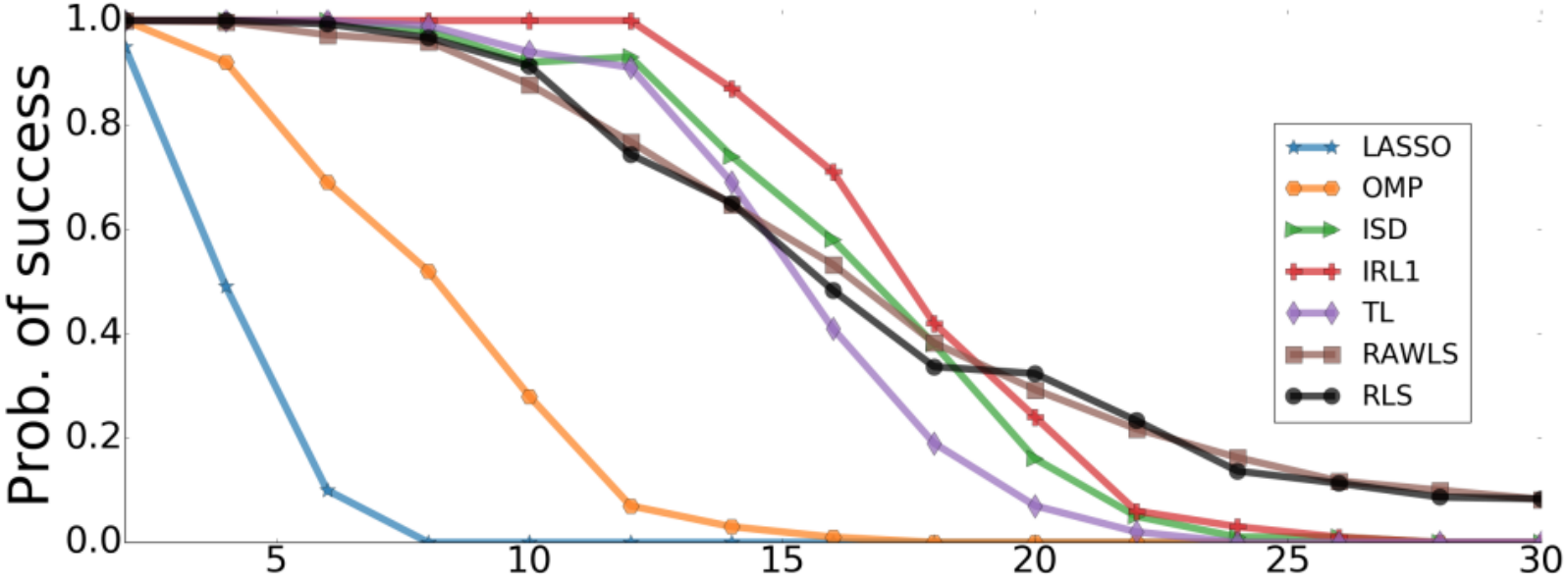
How do these methods compare?

64 unknown variables (say, genes) and $\theta \in \{-1, 0, 1\}^{64}$ has 30 nonzero entries (not that sparse) and $\mathcal{N}(0, 0.5)$ Gaussian noise. How many 'experiments' (equations) do you need to recover θ?



How do these methods compare?

64 unknown variables and $\mathcal{N}(0, 1)$ Gaussian noise. Success as a function of sparsity.



Refined Least Squares (RLS)

The goal of the rest of the talk is to motivate the essence behind Refined Least Squares (RLS).

Refined Least Squares (RLS)

The goal of the rest of the talk is to motivate the essence behind Refined Least Squares (RLS).

I will mainly emphasize the underlying new idea.

Refined Least Squares (RLS)

The goal of the rest of the talk is to motivate the essence behind Refined Least Squares (RLS).

> *I will mainly emphasize the underlying new idea. It seems very likely that the underlying idea can be used to boost many other methods (example later).*

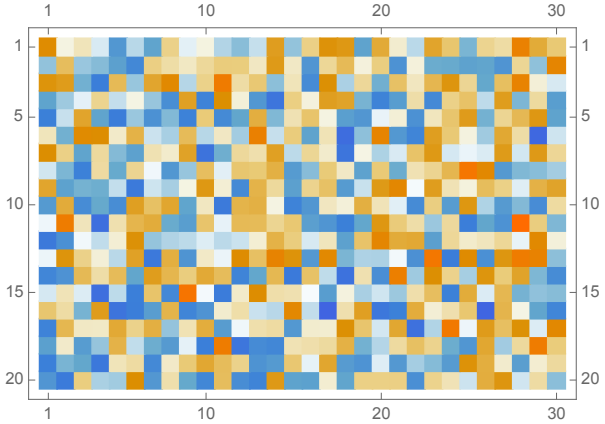
Refined Least Squares (RLS)

We take the same example as above. $A \in \mathbb{R}^{20\times 30}$ as above

Refined Least Squares (RLS)

We take the same example as above. $A \in \mathbb{R}^{20\times 30}$ as above

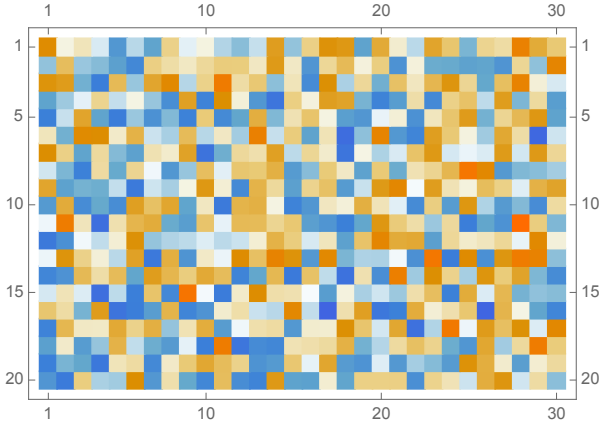
$$y = \quad \theta + \omega$$



where $\theta = (1, 1, 0, 0, \dots)$ and $\omega_i \sim \mathcal{N}(0, 1)$. How to get θ from y?

Refined Least Squares (RLS)

We take the same example as above. $A \in \mathbb{R}^{20\times 30}$ as above

$y =$  $\theta + \omega$

where $\theta = (1, 1, 0, 0, \dots)$ and $\omega_i \sim \mathcal{N}(0, 1)$. How to get θ from y?

Simplest possible idea. Let's just do least squares

$$\|y - Ax\|_2 \to \min .$$

Refined Least Squares (RLS)

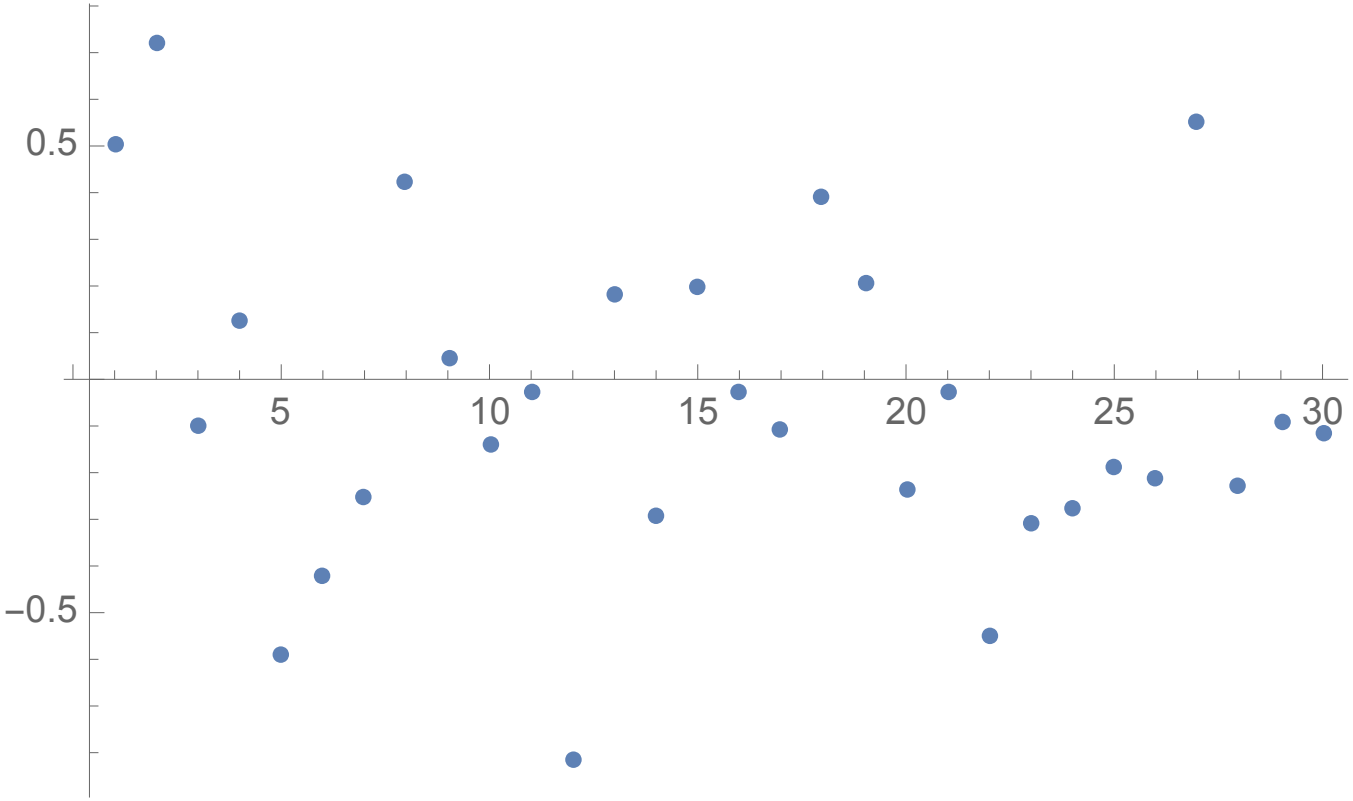
Let's just do least squares

$$\|y - Ax\|_2 \to \min .$$

Refined Least Squares (RLS)

Let's just do least squares

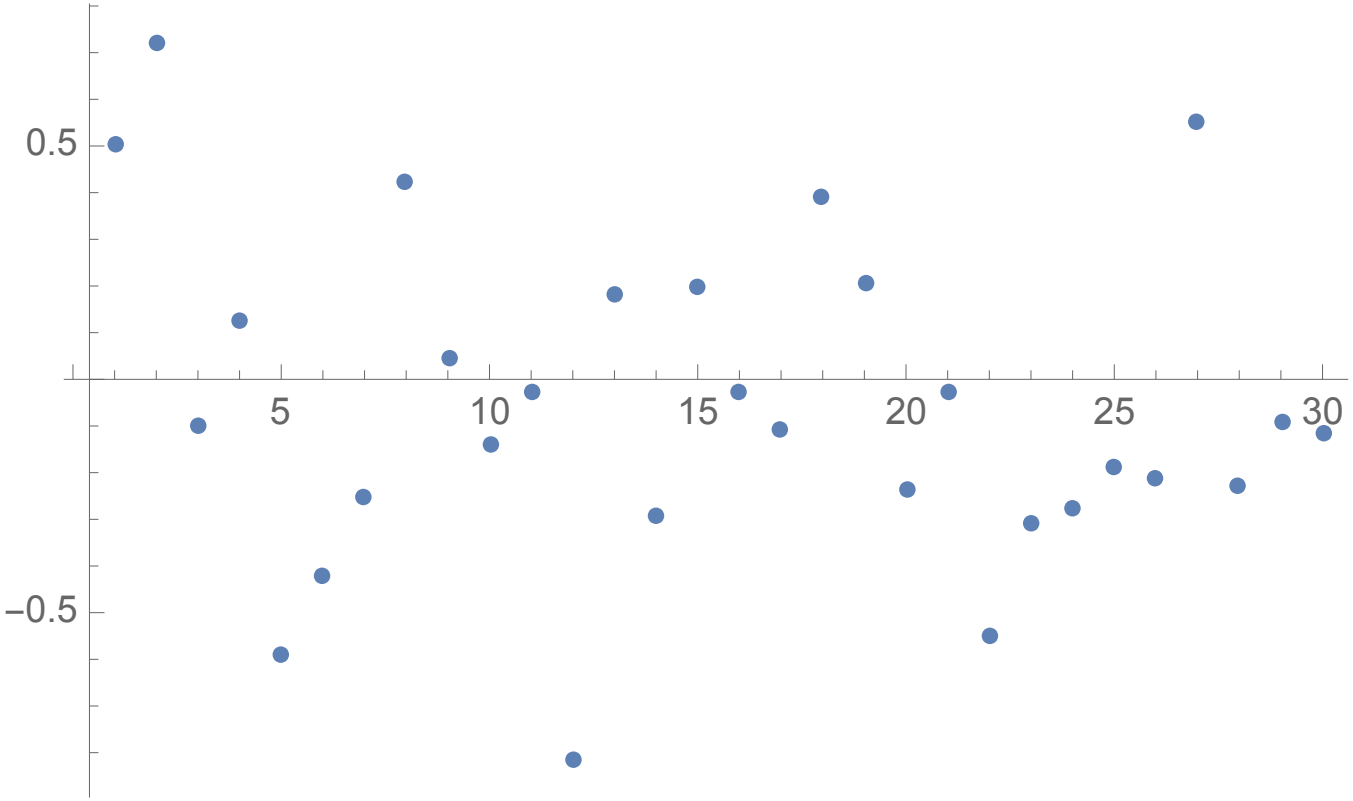
$$\|y - Ax\|_2 \to \min .$$



Refined Least Squares (RLS)

Let's just do least squares

$$\|y - Ax\|_2 \to \min .$$



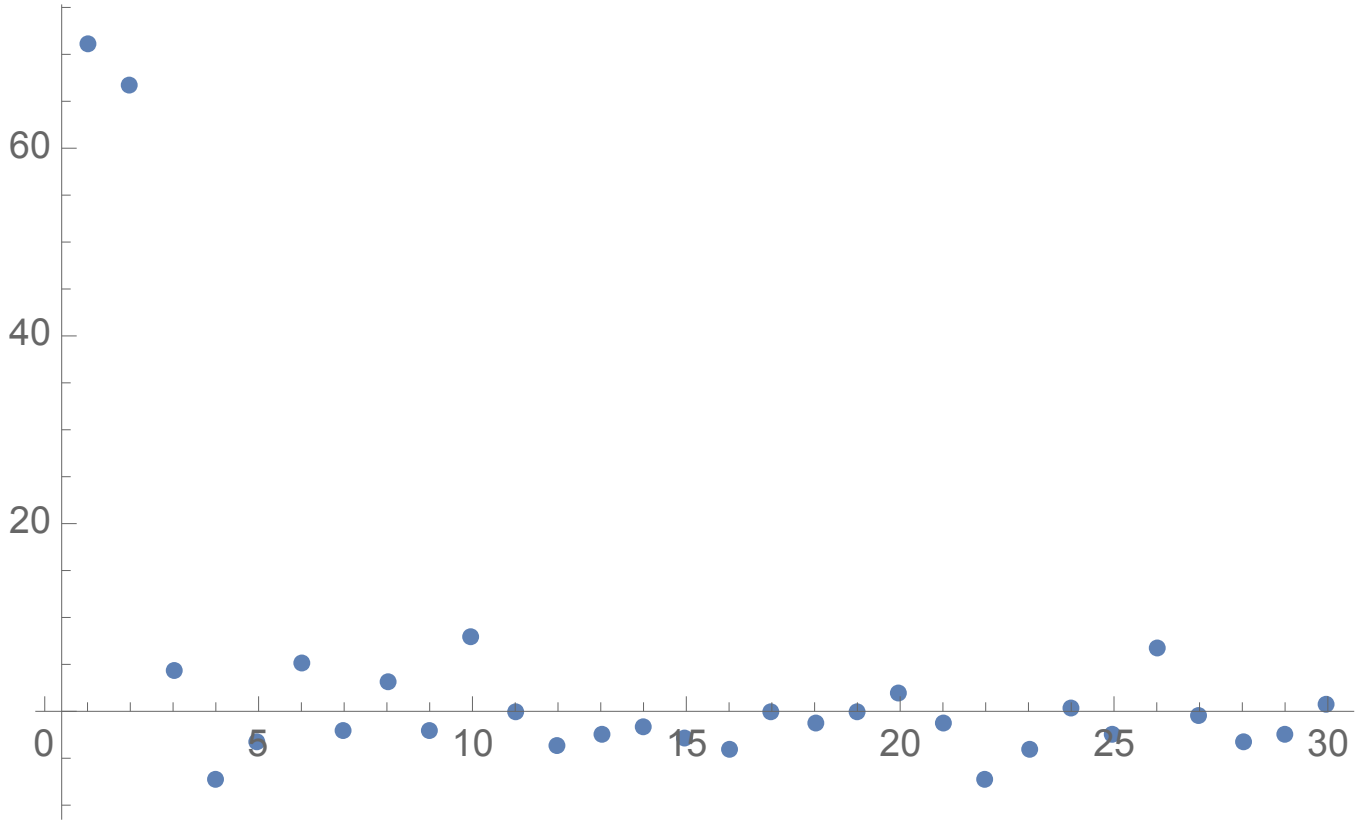
This is **very** bad. Certainly the first two coefficients are not small but many others are bigger.

Refined Least Squares (RLS)

Least Squares does not work – but it works *on average*. Let's take the same example, $\theta = (1, 1, 0, 0 \ldots, 0)$ and average the behavior of Least Squares over 100 random choices of A, ω.

Refined Least Squares (RLS)

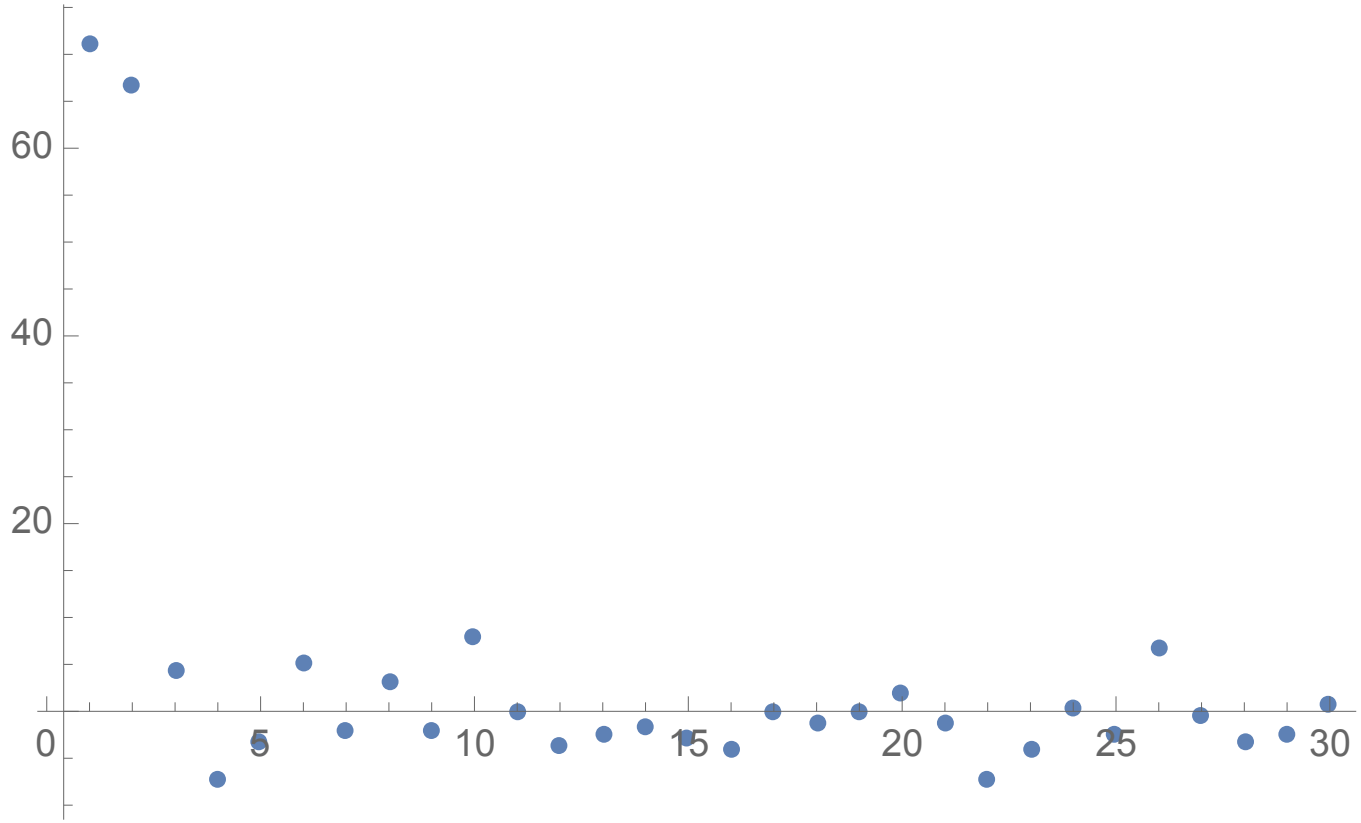
Least Squares does not work – but it works *on average*. Let's take the same example, $\theta = (1, 1, 0, 0 \ldots, 0)$ and average the behavior of Least Squares over 100 random choices of A, ω.



That works!

Refined Least Squares (RLS)

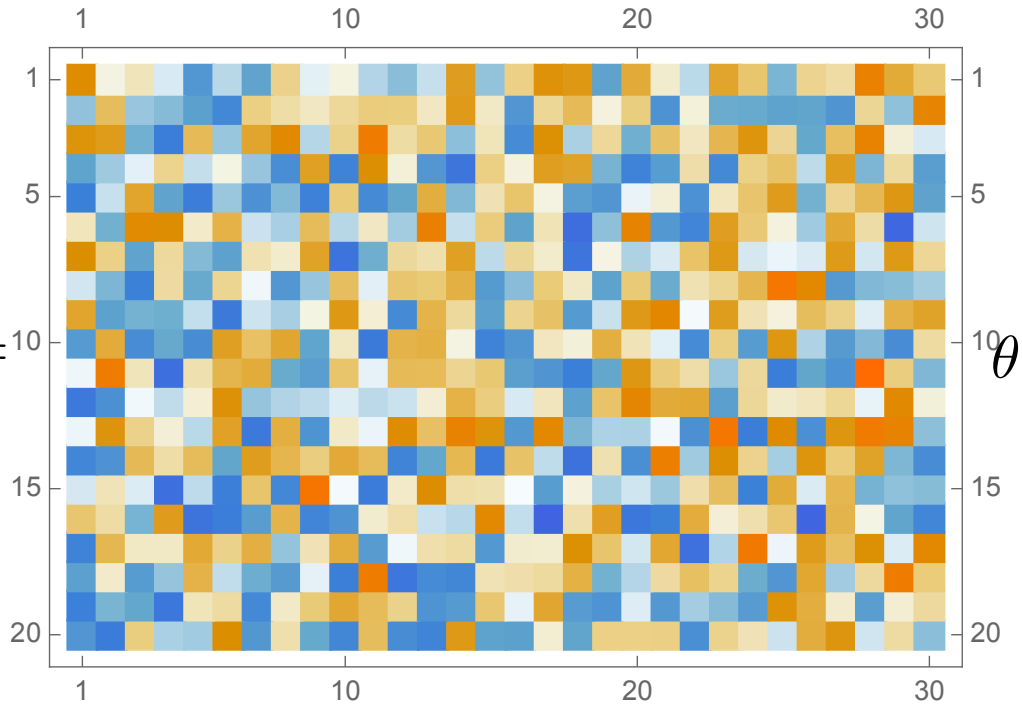
Least Squares does not work – but it works *on average*. Let's take the same example, $\theta = (1, 1, 0, 0 \ldots, 0)$ and average the behavior of Least Squares over 100 random choices of A, ω.



That works! But it's clearly cheating: 100 random matrices is like having 100 times the number of equations...

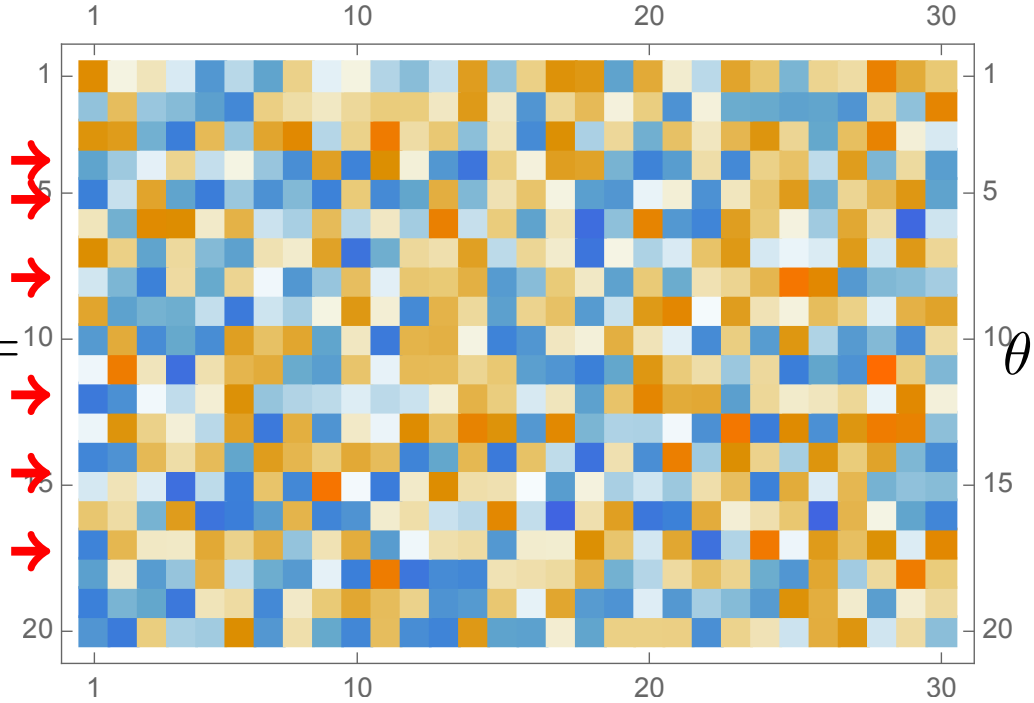
Refined Least Squares (RLS)

How to get more equations *without cheating*:

$y =$  $\theta + \omega$

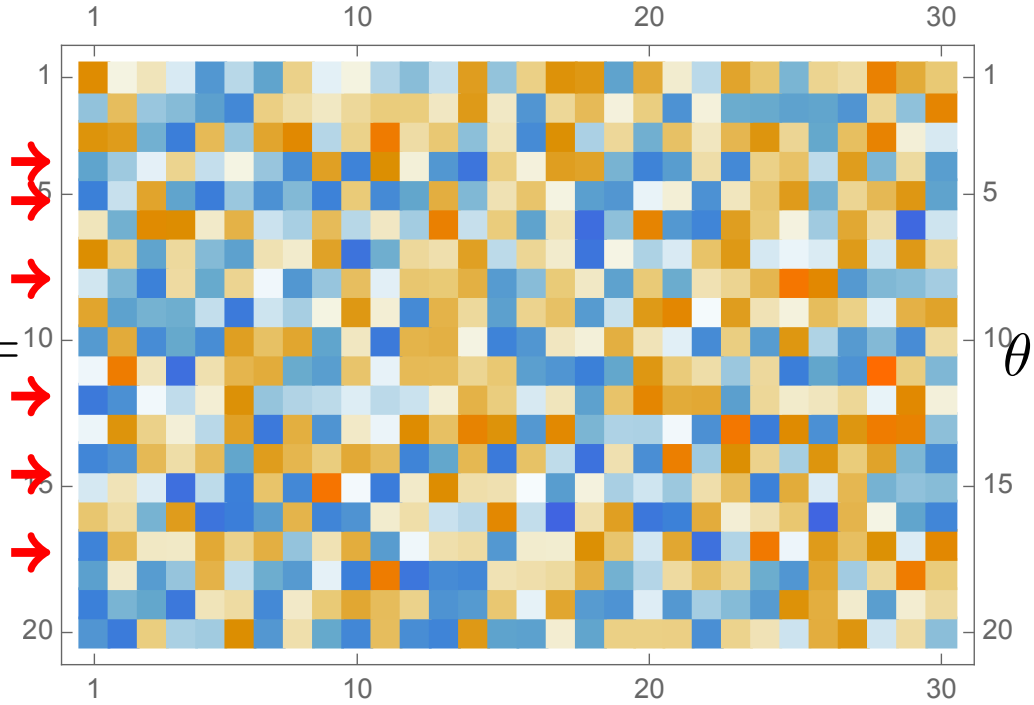
Refined Least Squares (RLS)

How to get more equations *without cheating*:

$y =$  $\theta + \omega$

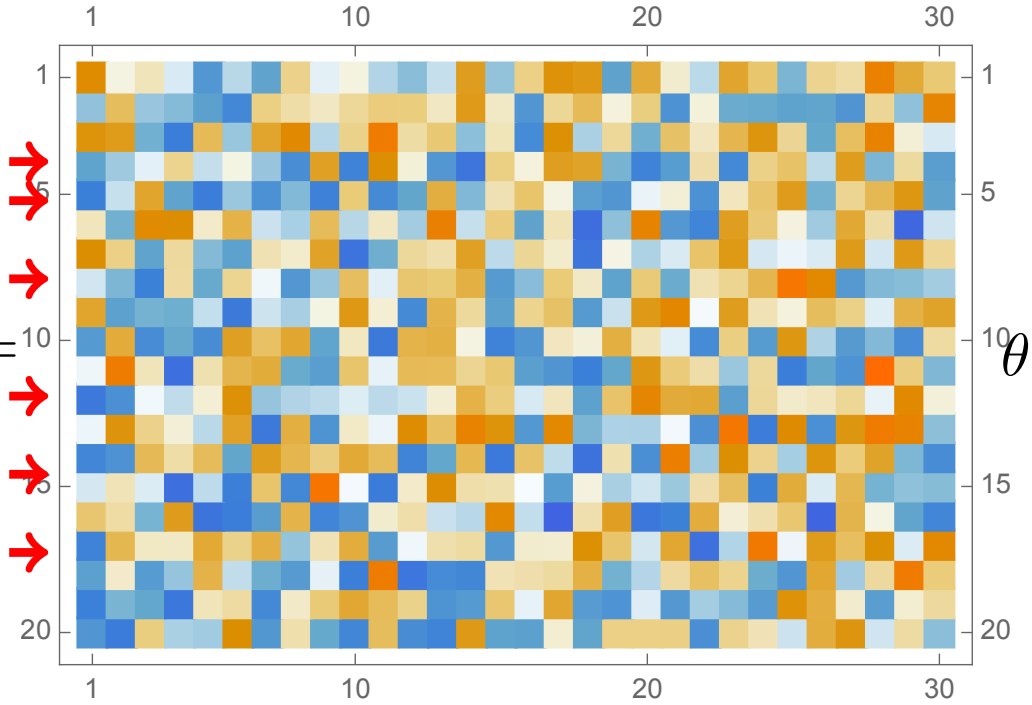
Just pick some random subset of rows: this gives you a '*new*' problem. It's easy to create many 'new' problems like this.

Refined Least Squares (RLS)

$y =$  $\theta + \omega$

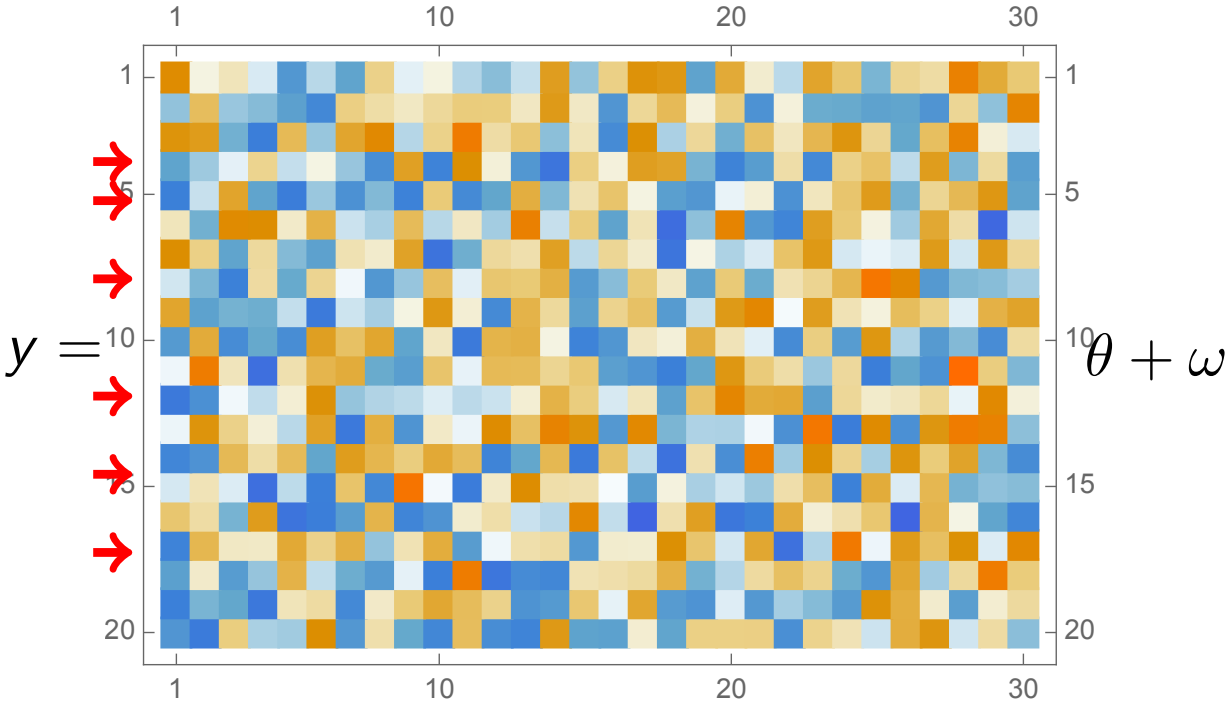
Let's do an example.

Refined Least Squares (RLS)

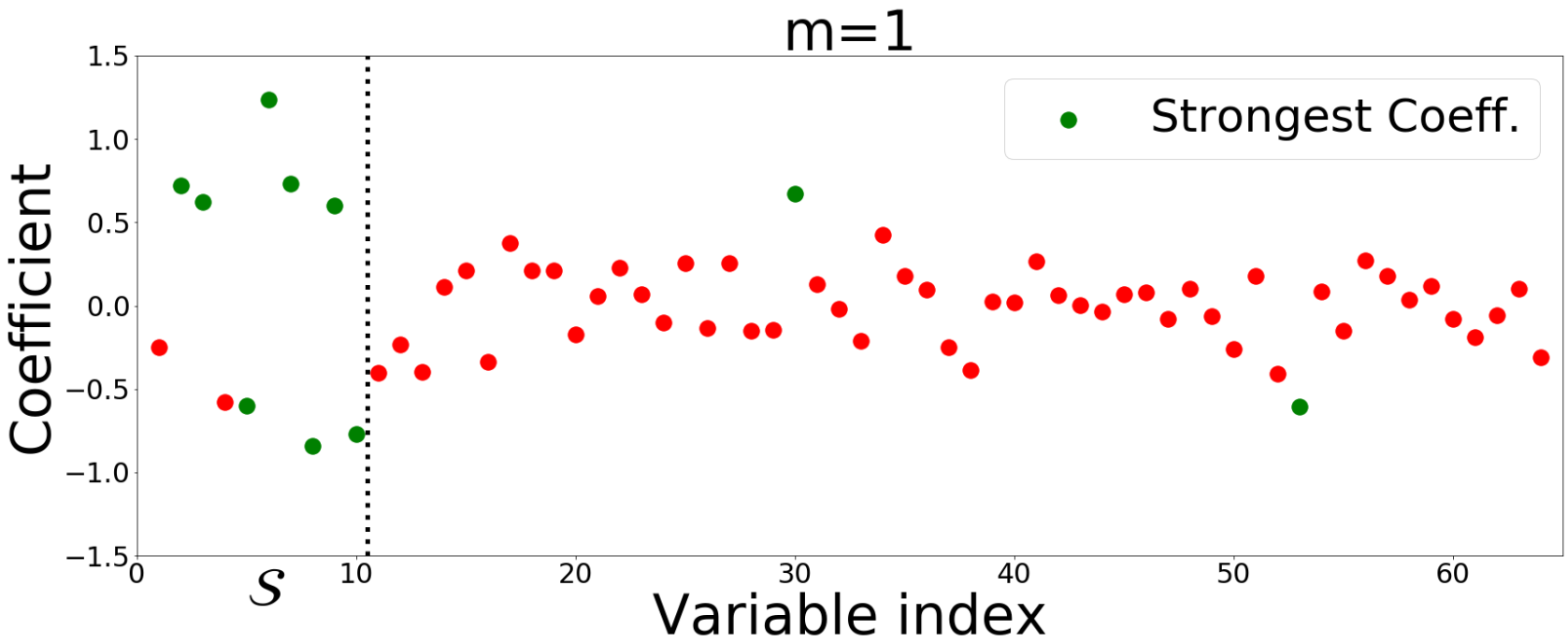
$y =$  $\theta + \omega$

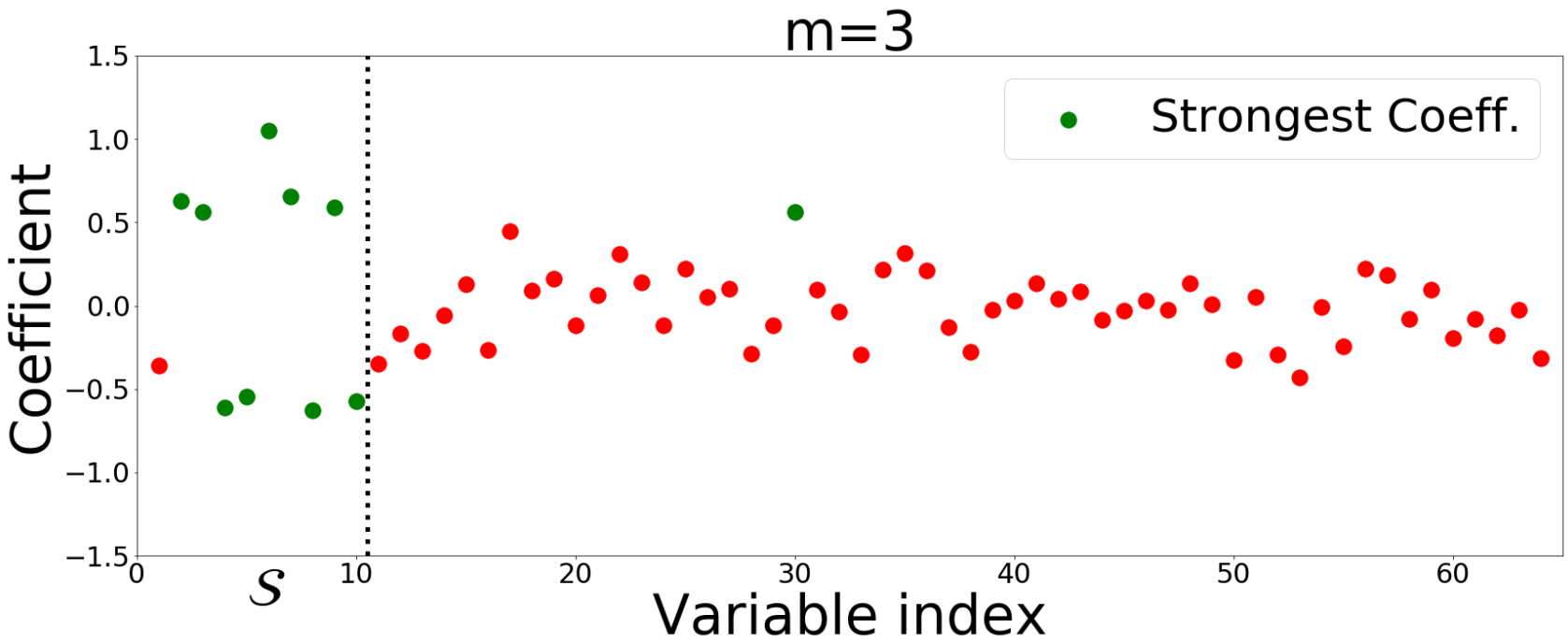
Let's do an example. 64 unknowns, the ground truth is supported in the first 10 coordinates.

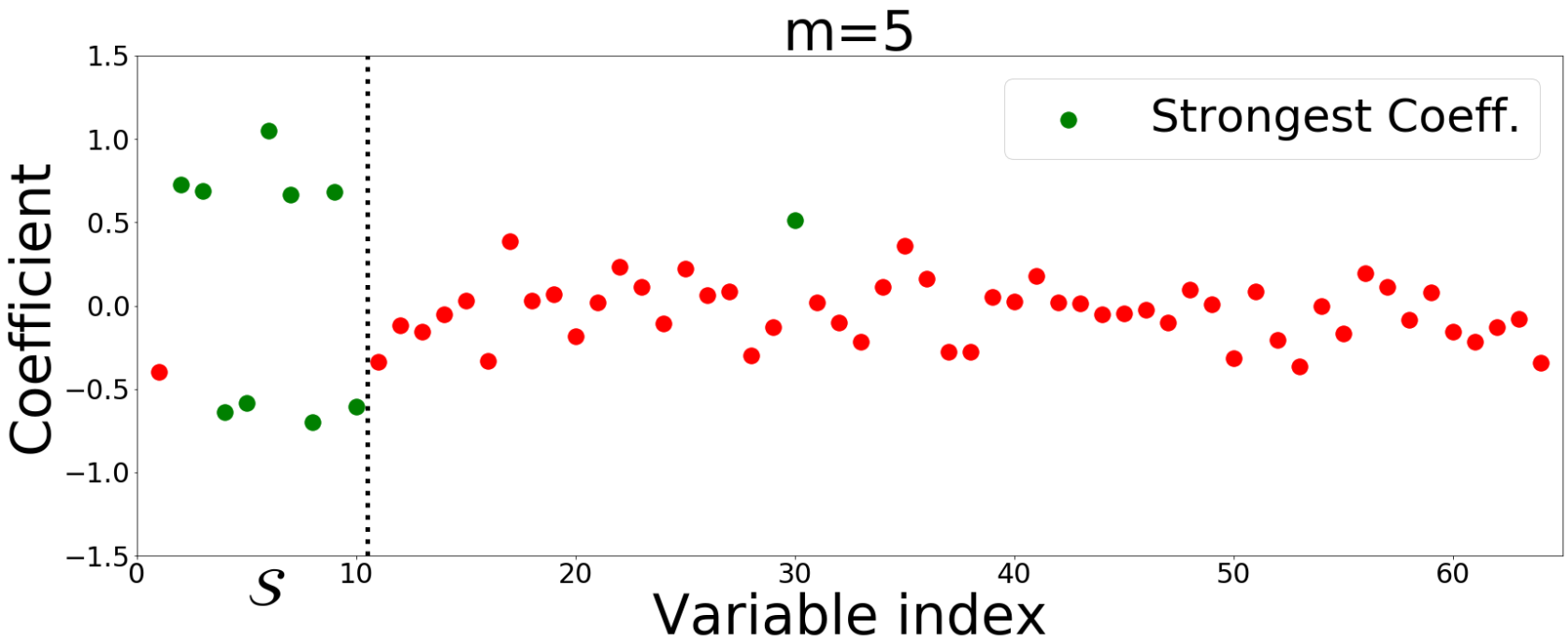
Refined Least Squares (RLS)

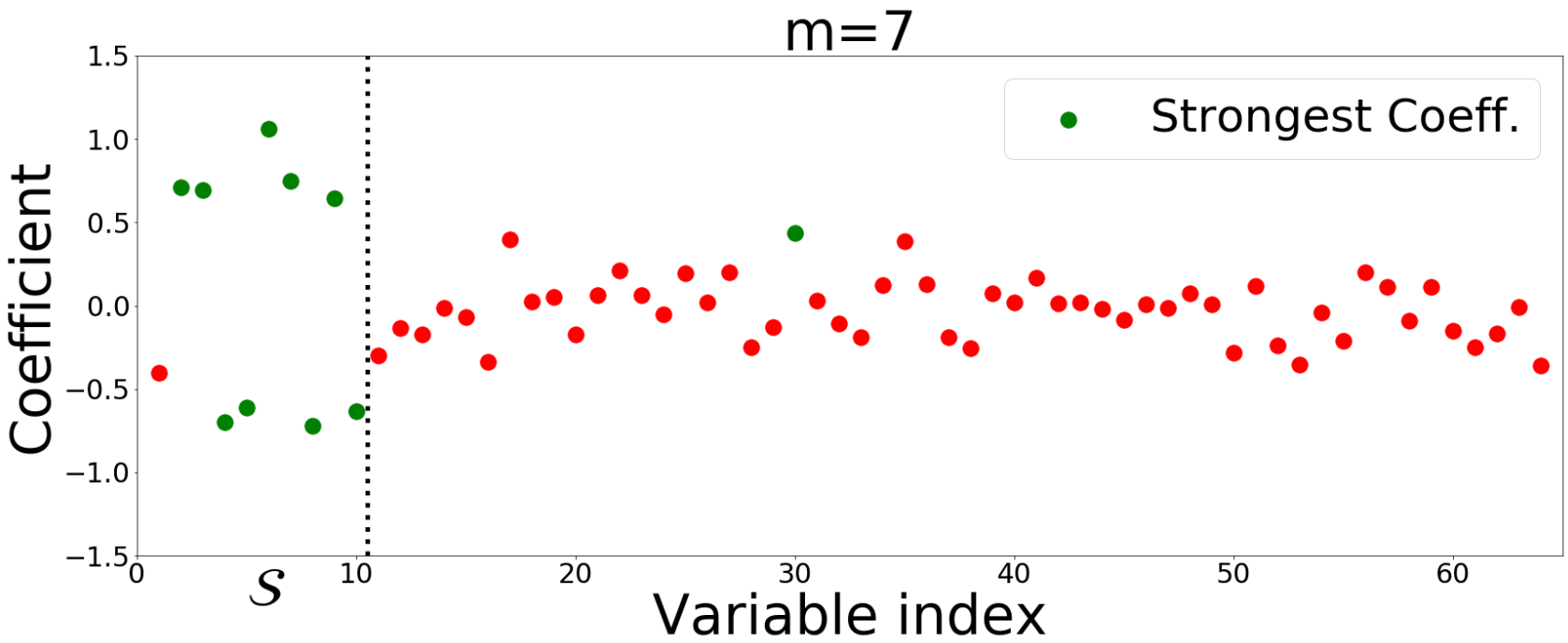


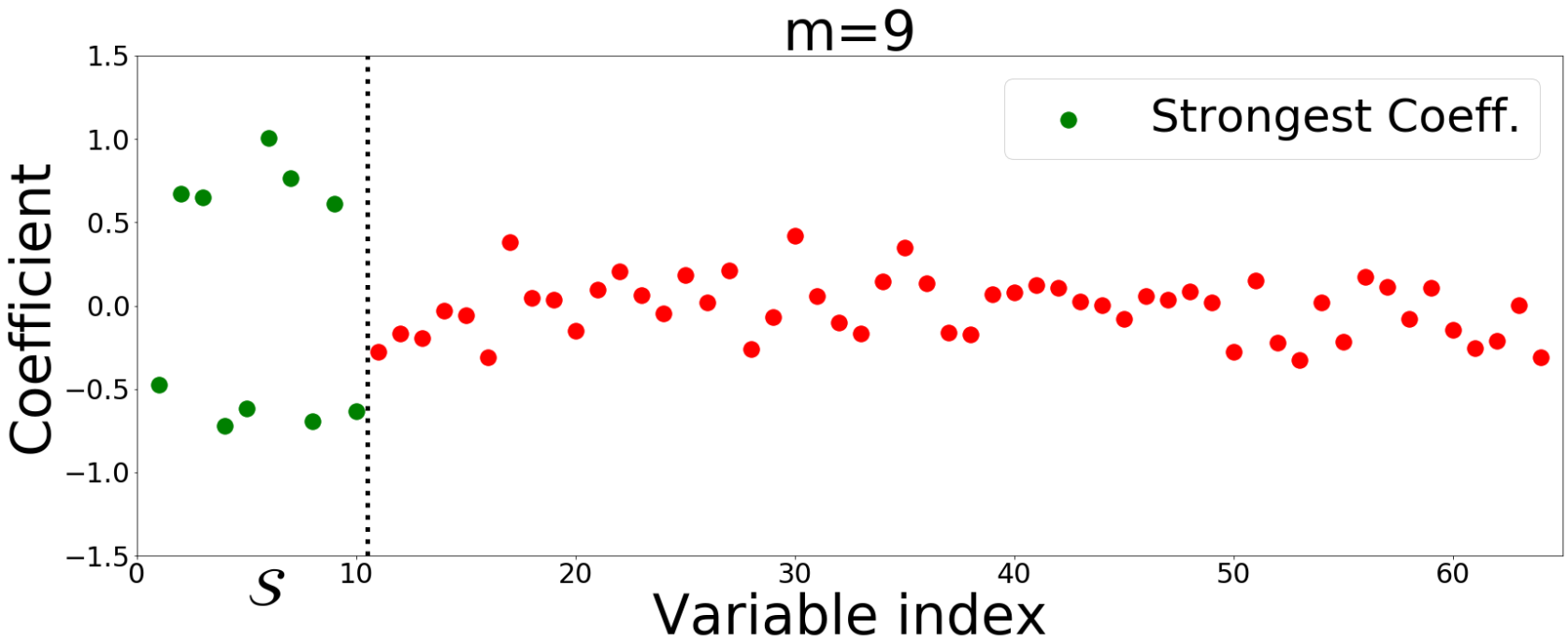
Let's do an example. 64 unknowns, the ground truth is supported in the first 10 coordinates. We average over m random subsets each of which take each row with likelihood $p \sim 0.6$.

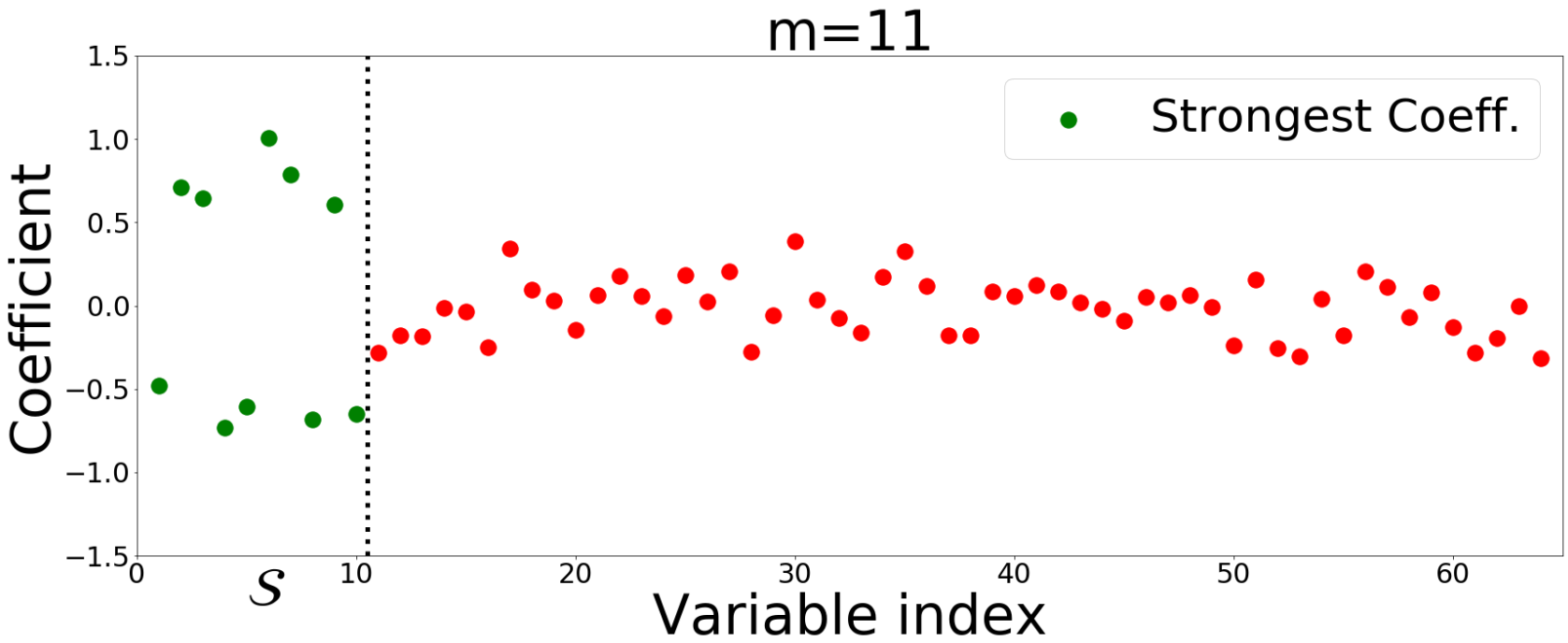

m=1
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index

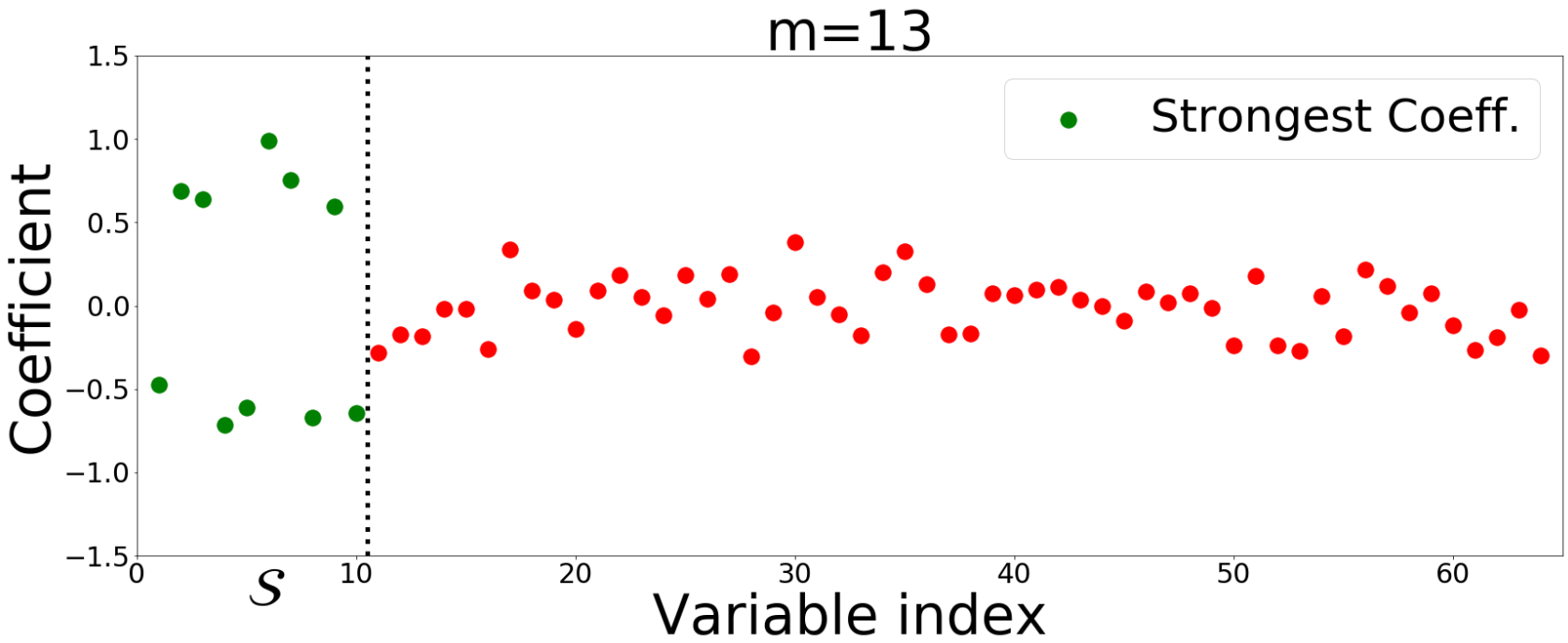

m=3
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index

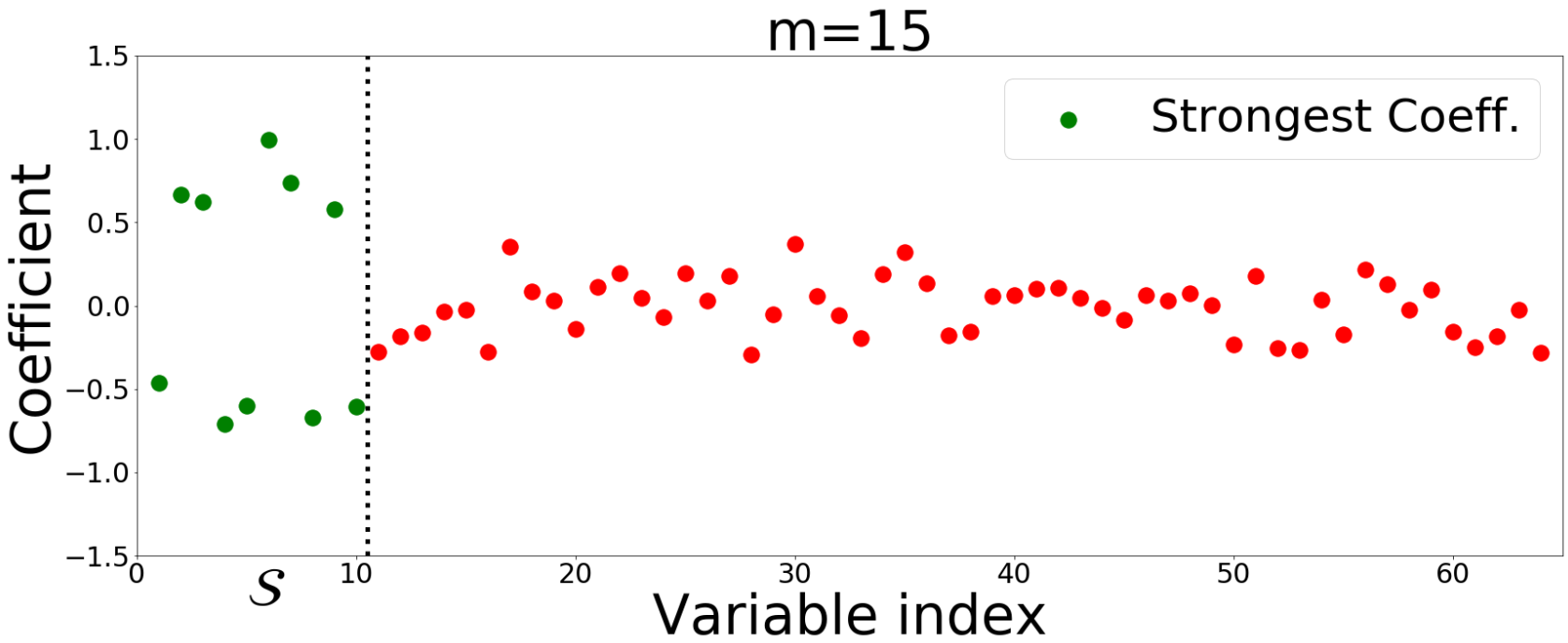

m=5
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index

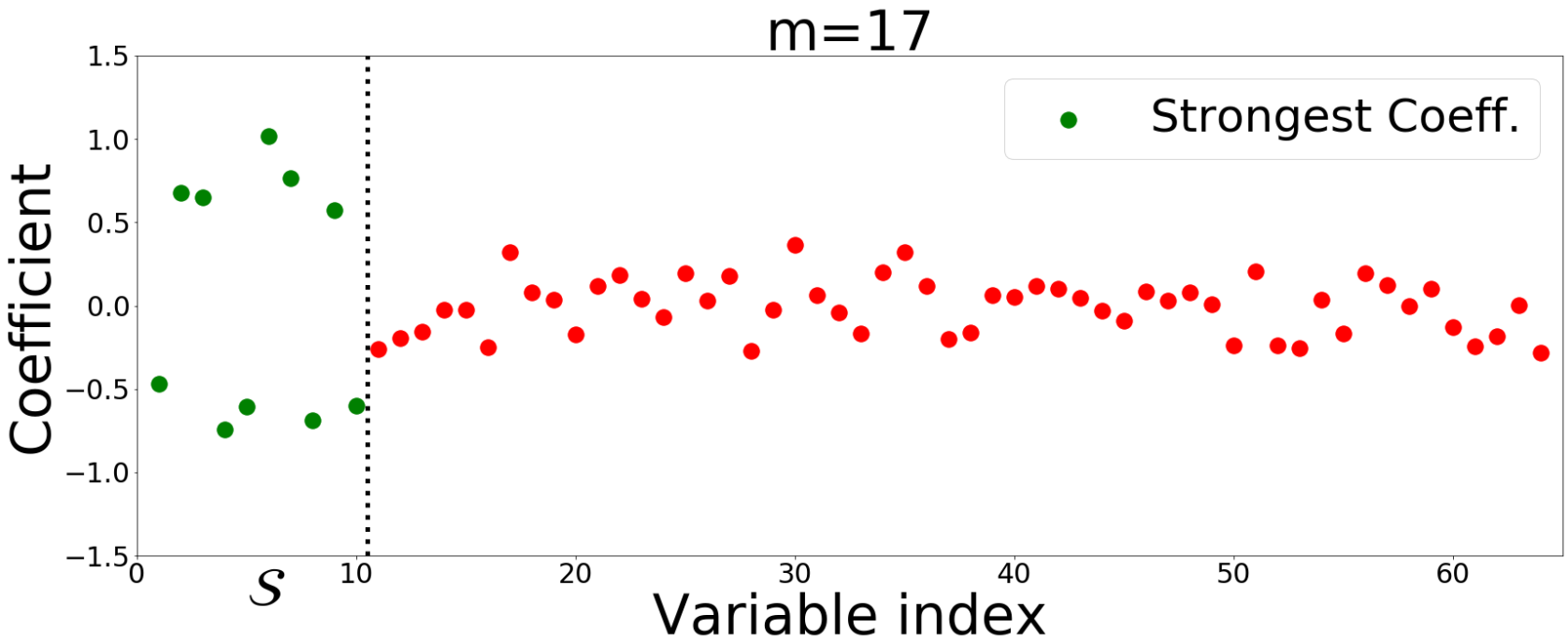

m=7
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index

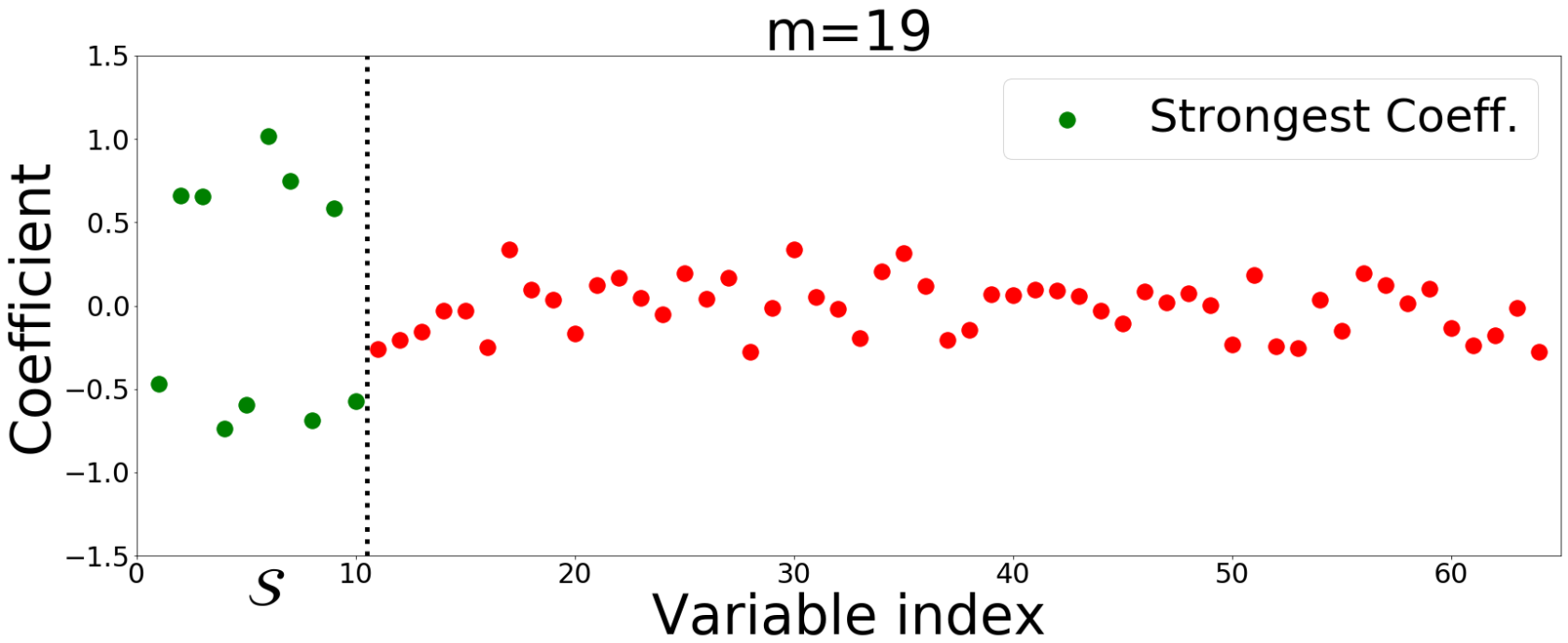

m=9
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index


m=11
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index


m=13
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index


m=15
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index


m=17
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index


m=19
Strongest Coeff.
Coefficient
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50
60
s
Variable index

Refined Least Squares (RLS)

Outline of the algorithm.

Refined Least Squares (RLS)

Outline of the algorithm.

1. Pick a random subset of the equations and solve the problem with least squares.

Refined Least Squares (RLS)

Outline of the algorithm.

1. Pick a random subset of the equations and solve the problem with least squares.
2. Average the results.

Refined Least Squares (RLS)

Outline of the algorithm.

1. Pick a random subset of the equations and solve the problem with least squares.
2. Average the results.
3. Pick the largest entry in the average as a guess for a coordinate with a nonzero entry. Use the sign as a guess for the sign.

Refined Least Squares (RLS)

Outline of the algorithm.

1. Pick a random subset of the equations and solve the problem with least squares.
2. Average the results.
3. Pick the largest entry in the average as a guess for a coordinate with a nonzero entry. Use the sign as a guess for the sign.
4. Remove the corresponding coordinate to get a new problem with the same number of equations and one less unknown.

Refined Least Squares (RLS)

Outline of the algorithm.

1. Pick a random subset of the equations and solve the problem with least squares.
2. Average the results.
3. Pick the largest entry in the average as a guess for a coordinate with a nonzero entry. Use the sign as a guess for the sign.
4. Remove the corresponding coordinate to get a new problem with the same number of equations and one less unknown.
5. Go back up to 1.

Refined Least Squares (RLS)

Outline of the algorithm.

1. Pick a random subset of the equations and solve the problem with least squares.
2. Average the results.
3. Pick the largest entry in the average as a guess for a coordinate with a nonzero entry. Use the sign as a guess for the sign.
4. Remove the corresponding coordinate to get a new problem with the same number of equations and one less unknown.
5. Go back up to 1.

Refined Least Squares (RLS)

Two small lies on the previous slide. One is ε^2 (we average over some parameters), the other one is more interesting.

Refined Least Squares (RLS)

Two small lies on the previous slide. One is ε^2 (we average over some parameters), the other one is more interesting.

We run the algorithm, detect candidates for the support and remove them.

Refined Least Squares (RLS)

Two small lies on the previous slide. One is ε^2 (we average over some parameters), the other one is more interesting.

We run the algorithm, detect candidates for the support and remove them. This means that we are forced to work with less and less signal, the noise remains constant.

Refined Least Squares (RLS)

Two small lies on the previous slide. One is ε^2 (we average over some parameters), the other one is more interesting.

We run the algorithm, detect candidates for the support and remove them. This means that we are forced to work with less and less signal, the noise remains constant.

At the very end, we face a familar problem:

Refined Least Squares (RLS)

Two small lies on the previous slide. One is ε^2 (we average over some parameters), the other one is more interesting.

We run the algorithm, detect candidates for the support and remove them. This means that we are forced to work with less and less signal, the noise remains constant.

At the very end, we face a familar problem: you have a list of vector $v_1, \ldots, v_n$ in front of you.

Refined Least Squares (RLS)

Two small lies on the previous slide. One is ε^2 (we average over some parameters), the other one is more interesting.

We run the algorithm, detect candidates for the support and remove them. This means that we are forced to work with less and less signal, the noise remains constant.

At the very end, we face a familar problem: you have a list of vector $v_1, \ldots, v_n$ in front of you. Somebody gives you

$$y = v_i + \omega.$$

How do you have the best chance of getting v_i?

Refined Least Squares (RLS)

Two small lies on the previous slide. One is ε^2 (we average over some parameters), the other one is more interesting.

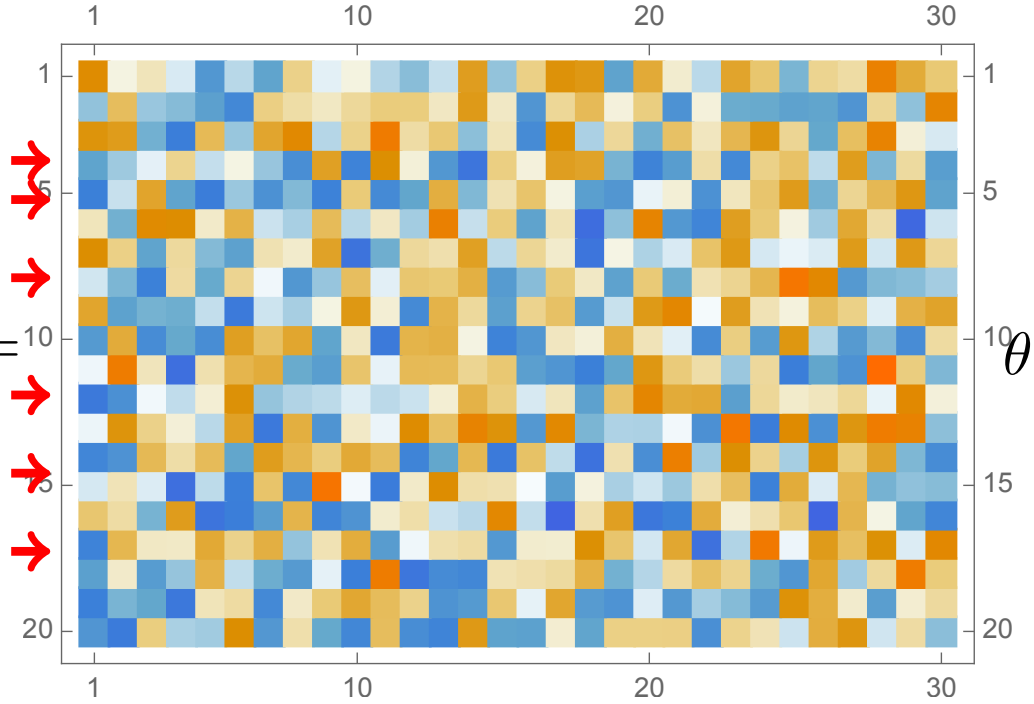
We run the algorithm, detect candidates for the support and remove them. This means that we are forced to work with less and less signal, the noise remains constant.

At the very end, we face a familar problem: you have a list of vector $v_1, \ldots, v_n$ in front of you. Somebody gives you

$$y = v_i + \omega.$$

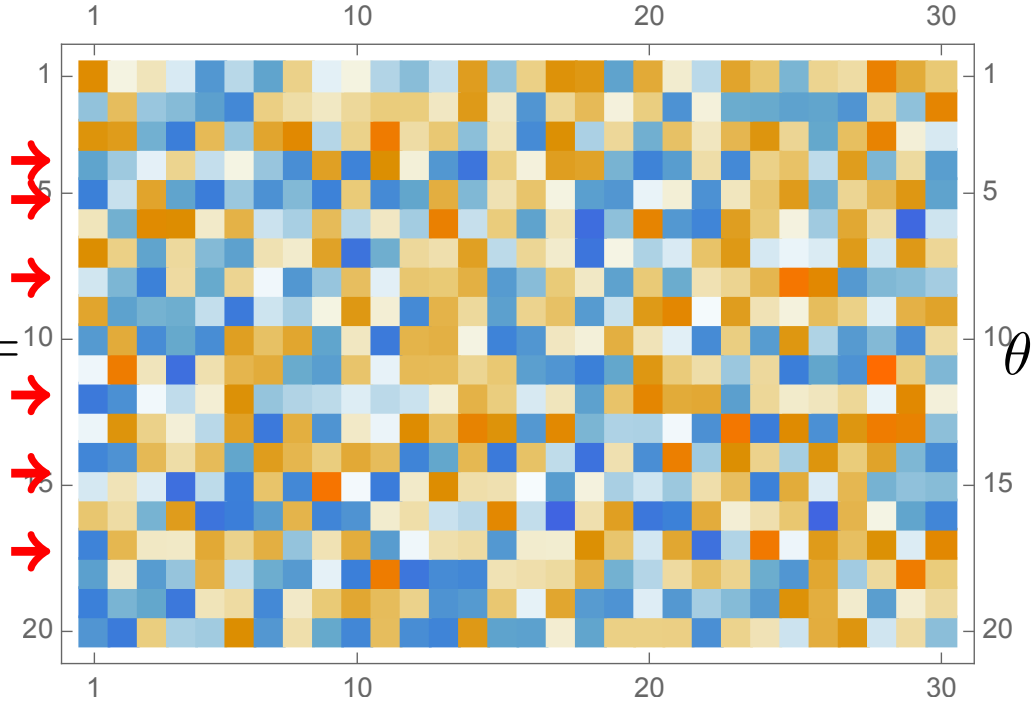
How do you have the best chance of getting v_i? At this stage, we switch back to Inner Products (the OMP selection rule).

Refined Least Squares (RLS)

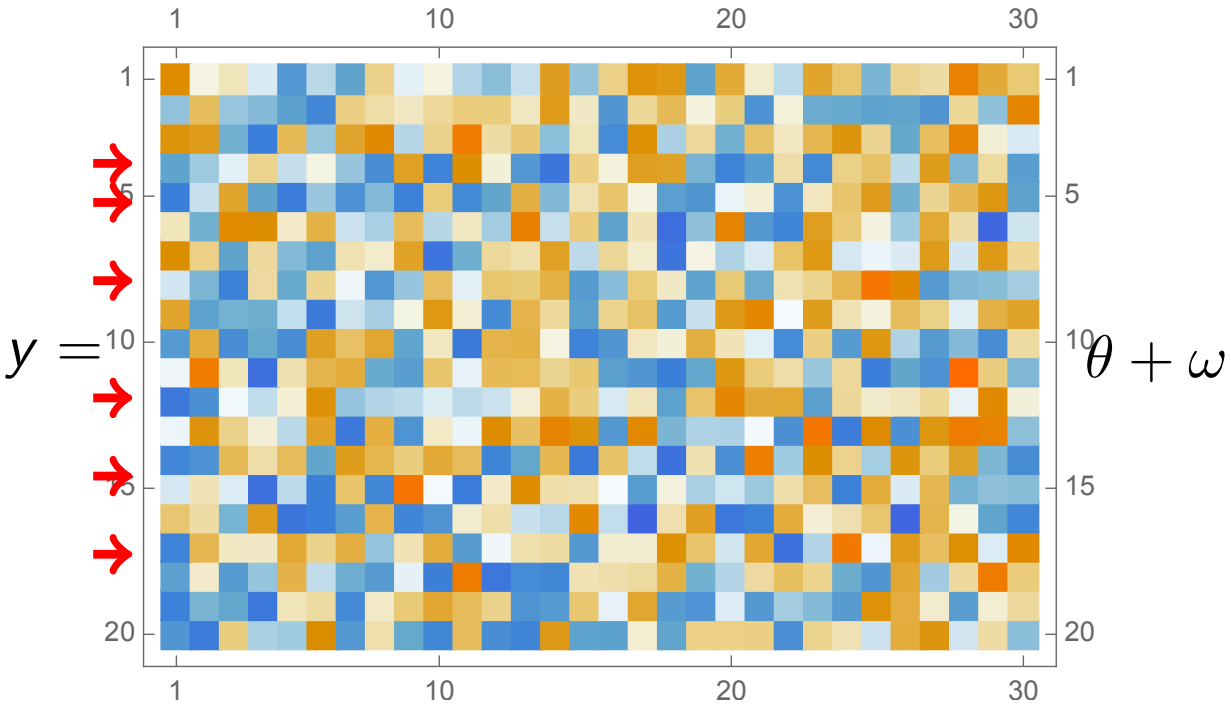
$y =$  $\theta + \omega$

Underlying idea: by looking at subsets of the equations, we get a harder problem.

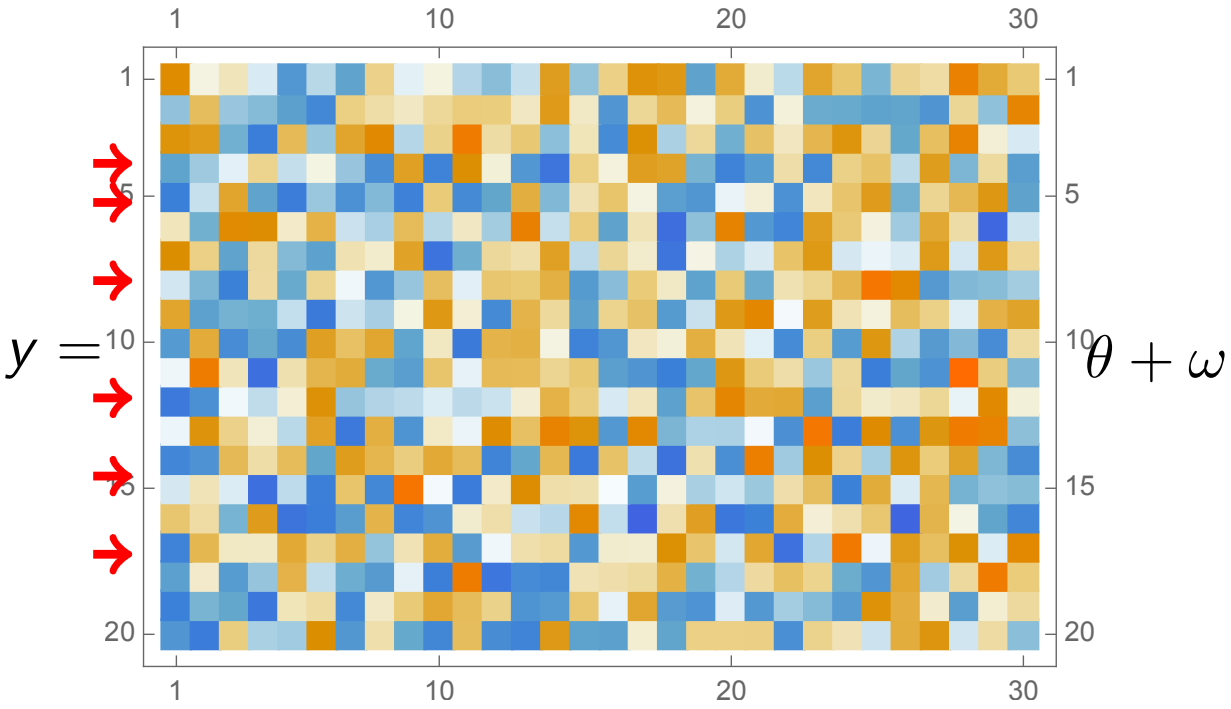
Refined Least Squares (RLS)

$y =$  $\theta + \omega$

Underlying idea: by looking at subsets of the equations, we get a harder problem. However, we get many harder problems and the gain from being able to average exceeds the increase of difficulty.



The main problem is so underdetermined that the randomly selected subproblems are relatively independent even though they share similar equations



The main problem is so underdetermined that the randomly selected subproblems are relatively independent even though they share similar equations

and this principle can be implied to other methods as well.

Boosting classical OMP

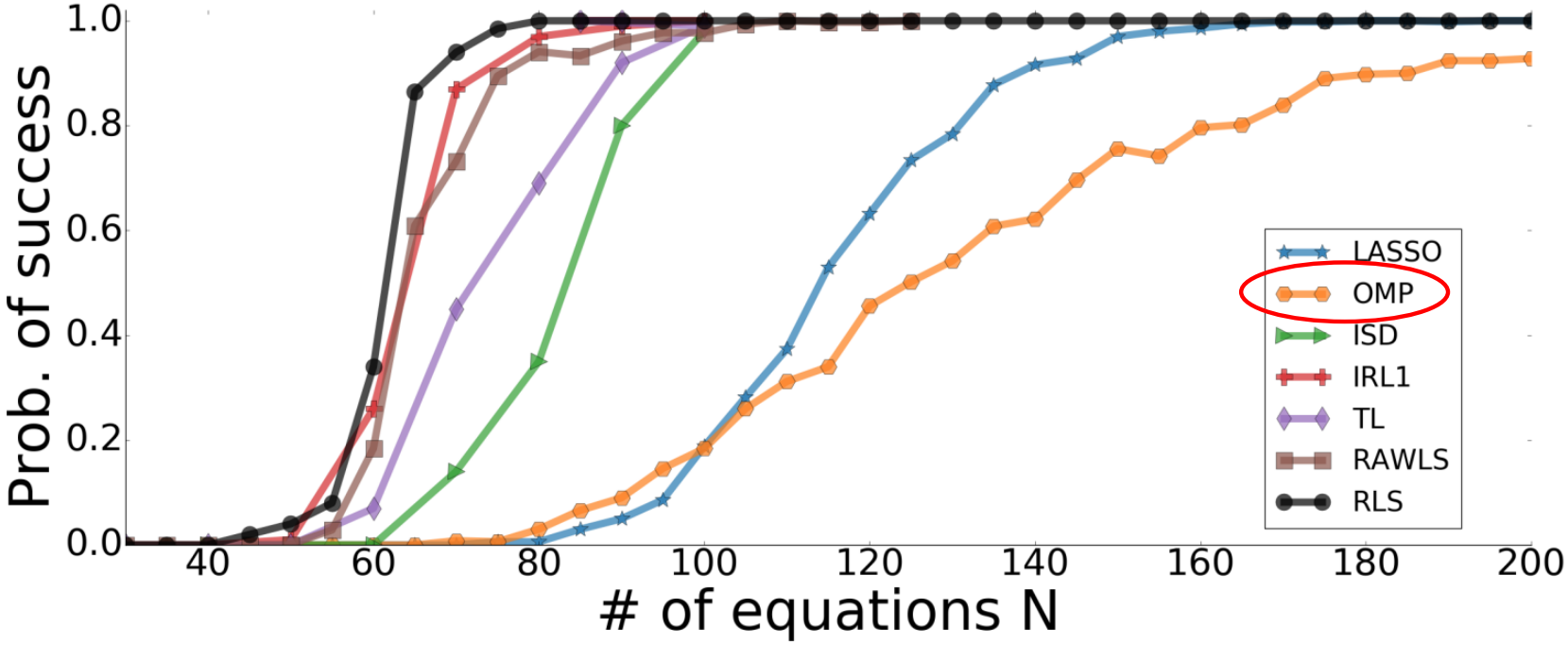
Classical OMP. Take inner product of RHS with columns.

Boosting classical OMP

Classical OMP. Take inner product of RHS with columns.
Assume largest inner product corresponds to a signal, remove it.

Boosting classical OMP

Classical OMP. Take inner product of RHS with columns.
Assume largest inner product corresponds to a signal, remove it.



Boosting classical OMP

Classical OMP. Take inner product of RHS with columns.
Assume largest inner product corresponds to a signal, remove it.

Boosted OMP. Pick a random subsets of the equations.

Boosting classical OMP

Classical OMP. Take inner product of RHS with columns. Assume largest inner product corresponds to a signal, remove it.

Boosted OMP. Pick a random subsets of the equations. For this reduced problem, create the vector

$$(\langle \text{(reduced) RHS}, \text{(reduced) column}_i \rangle)_{i=1}^{\#\text{columns}}.$$

Boosting classical OMP

Classical OMP. Take inner product of RHS with columns. Assume largest inner product corresponds to a signal, remove it.

Boosted OMP. Pick a random subsets of the equations. For this reduced problem, create the vector

$$(\langle(\text{reduced})\ \text{RHS}, (\text{reduced})\ \text{column}_i\rangle)_{i=1}^{\#\text{columns}}.$$

Average over many such vectors and then proceed as above.

Boosting classical OMP

Classical OMP. Take inner product of RHS with columns. Assume largest inner product corresponds to a signal, remove it.

Boosted OMP. Pick a random subsets of the equations. For this reduced problem, create the vector

$$(\langle(\text{reduced})\ \text{RHS}, (\text{reduced})\ \text{column}_i\rangle)_{i=1}^{\#\text{columns}}.$$

Average over many such vectors and then proceed as above.

Experiments suggests that this boosts OMP to a **very competitive** method (it seems slightly worse than RLS but only slightly).

Some Theory

Here is a basic toy model. We have

$$y = X\theta + \omega,$$

where $X \in \mathbb{R}^{N \times D}$ is a (standard) Gaussian random matrix and ω is (standard) Gaussian noise.

Some Theory

Here is a basic toy model. We have

$$y = X\theta + \omega,$$

where $X \in \mathbb{R}^{N \times D}$ is a (standard) Gaussian random matrix and ω is (standard) Gaussian noise. How far is the least squares approximation from the ground truth – how does least squares handle the random error?

Some Theory

Here is a basic toy model. We have

$$y = X\theta + \omega,$$

where $X \in \mathbb{R}^{N \times D}$ is a (standard) Gaussian random matrix and ω is (standard) Gaussian noise. How far is the least squares approximation from the ground truth – how does least squares handle the random error?

Proposition (Lindenbaum, S, 21)

If we fix the ratio $N/D < 1$ and let the dimensions of the matrix go to infinity, then

$$\mathbb{E}_{X,\omega} \| X^\dagger y - X^\dagger X \theta^* \| = (1 + o(1)) \sqrt{\frac{N}{D - N}}.$$

Some Theory

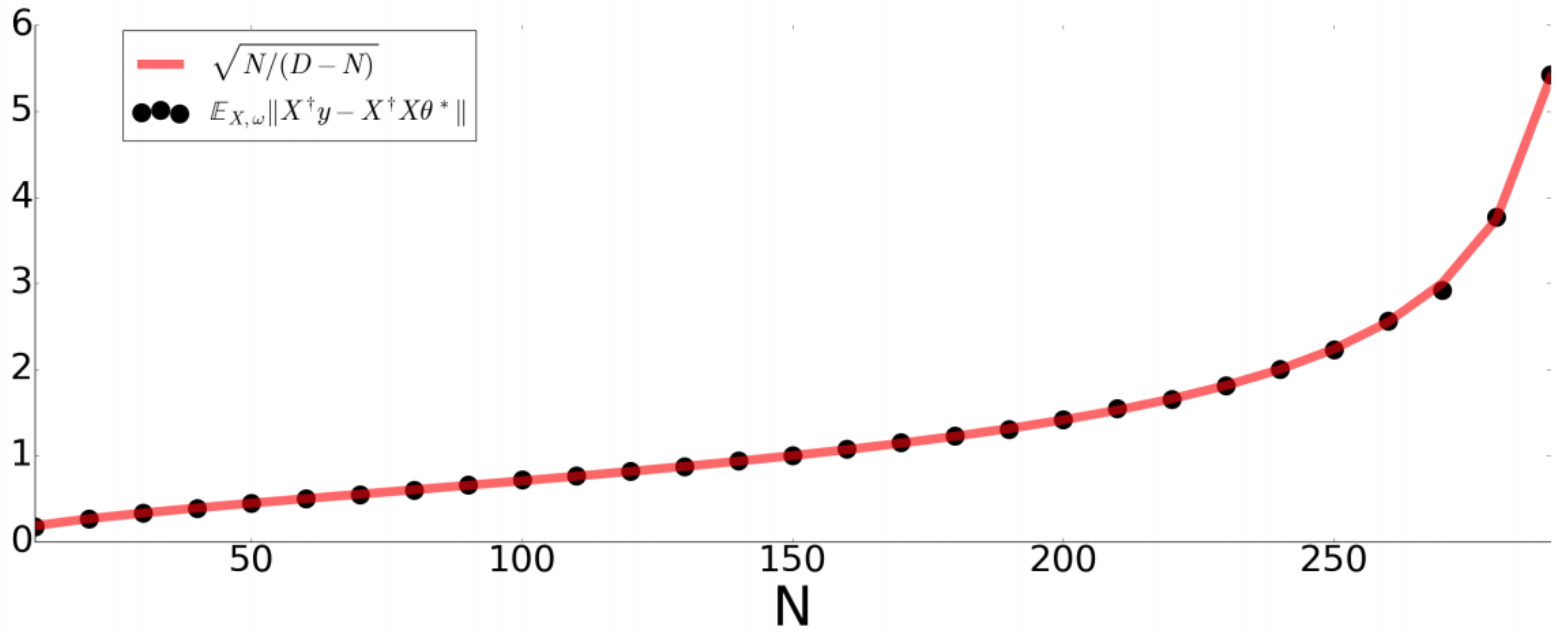


FIGURE 3. Numerical evaluation of the relation predicted by Theorem 1 for $D = 300$ and $10 \leq N \leq 290$. As the number of equations N tends to D (the number of variables), the expected ℓ^2 norm of $\boldsymbol{X}^\dagger \boldsymbol{y} - \boldsymbol{X}^\dagger \boldsymbol{X}\boldsymbol{\theta}^*$ (black dots) grows like $N^{1/2}(D-N)^{-1/2}$ (red line).

Some Theory

Proposition (Lindenbaum, S, 21)

If we fix the ratio $N/D < 1$ and let the dimensions of the matrix go to infinity, then

$$\mathbb{E}_{X,\omega}\|X^\dagger y - X^\dagger X\theta^*\| = (1 + o(1))\sqrt{\frac{N}{D-N}}.$$

When the matrix is very underdetermined, then a lot of the error gets lost in the projection.

Some Theory

Proposition (Lindenbaum, S, 21)

If we fix the ratio $N/D < 1$ and let the dimensions of the matrix go to infinity, then

$$\mathbb{E}_{X,\omega}\|X^\dagger y - X^\dagger X\theta^*\| = (1 + o(1))\sqrt{\frac{N}{D-N}}.$$

When the matrix is very underdetermined, then a lot of the error gets lost in the projection. As the matrix gets closer to square, the error sticks around.

Some Theory

Proposition (Lindenbaum, S, 21)

If we fix the ratio $N/D < 1$ and let the dimensions of the matrix go to infinity, then

$$\mathbb{E}_{X,\omega}\|X^\dagger y - X^\dagger X\theta^*\| = (1 + o(1))\sqrt{\frac{N}{D-N}}.$$

When the matrix is very underdetermined, then a lot of the error gets lost in the projection. As the matrix gets closer to square, the error sticks around. Also: almost square random Gaussian matrices have small singular values: the error is blown up.

Some Theory

Proposition (Lindenbaum, S, 21)

If we fix the ratio $N/D < 1$ and let the dimensions of the matrix go to infinity, then

$$\mathbb{E}_{X,\omega}\|X^\dagger y - X^\dagger X\theta^*\| = (1 + o(1))\sqrt{\frac{N}{D-N}}.$$

Proof.

A fun computation: relevant are the inverse singular values of X.

Some Theory

Proposition (Lindenbaum, S, 21)

If we fix the ratio $N/D < 1$ and let the dimensions of the matrix go to infinity, then

$$\mathbb{E}_{X,\omega}\|X^\dagger y - X^\dagger X\theta^*\| = (1 + o(1))\sqrt{\frac{N}{D-N}}.$$

Proof.

A fun computation: relevant are the inverse singular values of X. Use the Marchenko-Pastur distribution.

Some Theory

Proposition (Lindenbaum, S, 21)

If we fix the ratio $N/D < 1$ and let the dimensions of the matrix go to infinity, then

$$\mathbb{E}_{X,\omega}\|X^\dagger y - X^\dagger X\theta^*\| = (1+o(1))\sqrt{\frac{N}{D-N}}.$$

Proof.

A fun computation: relevant are the inverse singular values of X. Use the Marchenko-Pastur distribution. One integral

$$\begin{aligned}
&\int \frac{\sqrt{(\lambda_+ - x)(x-\lambda_-)}}{x^2}dx = \\
&= \frac{1}{x}\sqrt{-\lambda^2-(x-1)^2+2\lambda(1+x)} \\
&+ \arctan\left(\frac{1+\lambda-x}{\sqrt{-\lambda^2-(x-1)^2+2\lambda(1+x)}}\right) \\
&- \frac{1+\lambda}{1-\lambda}\arctan\left(\frac{x+\lambda(2+x)-\lambda^2-1}{(\lambda-1)\sqrt{\lambda^2+2\lambda(1+x)-(1+x)^2}}\right).
\end{aligned}$$

Some Theory

Another theoretical perspective: we are given

$$y = X\theta^* + \omega,$$

where $X \in \mathbb{R}^{N \times D}$.

Some Theory

Another theoretical perspective: we are given

$$y = X\theta^* + \omega,$$

where $X \in \mathbb{R}^{N \times D}$.

Theorem (Lindenbaum, S, 2020)

Averaging over random subsets A of the equations of size $n < 0.9D$, we have

$$\mathbb{E}_{X,\omega} \left\| \mathbb{E}_A \left(\pi_A \theta^* - \widehat{\theta}_A \right) \right\| \lesssim \frac{n}{\sqrt{N}\sqrt{D}} + \frac{n}{D}.$$

Some Theory

Theorem (Lindenbaum, S, 2020)

Averaging over random subsets A of the equations of size $n < 0.9D$, we have

$$\mathbb{E}_{X,\omega} \left\| \mathbb{E}_A \left(\pi_A \theta^* - \widehat{\theta}_A \right) \right\| \lesssim \frac{n}{\sqrt{N}\sqrt{D}} + \frac{n}{D}.$$

Some Theory

Theorem (Lindenbaum, S, 2020)

Averaging over random subsets A of the equations of size $n < 0.9D$, we have

$$\mathbb{E}_{X,\omega} \left\| \mathbb{E}_A \left(\pi_A \theta^* - \widehat{\theta}_A \right) \right\| \lesssim \frac{n}{\sqrt{N}\sqrt{D}} + \frac{n}{D}.$$

This leads to a dichotomy:

Some Theory

Theorem (Lindenbaum, S, 2020)

Averaging over random subsets A of the equations of size $n < 0.9D$, we have

$$\mathbb{E}_{X,\omega} \left\| \mathbb{E}_A \left(\pi_A \theta^* - \widehat{\theta}_A \right) \right\| \lesssim \frac{n}{\sqrt{N}\sqrt{D}} + \frac{n}{D}.$$

This leads to a dichotomy:

1. If n is small, then the error is small but the projection $\pi_A \theta^*$ has little to do with the ground truth.

Some Theory

Theorem (Lindenbaum, S, 2020)

Averaging over random subsets A of the equations of size $n < 0.9D$, we have

$$\mathbb{E}_{X,\omega} \left\| \mathbb{E}_A \left(\pi_A \theta^* - \widehat{\theta}_A \right) \right\| \lesssim \frac{n}{\sqrt{N}\sqrt{D}} + \frac{n}{D}.$$

This leads to a dichotomy:

1. If n is small, then the error is small but the projection $\pi_A \theta^*$ has little to do with the ground truth.
2. If n is large, then the error increases but the projection is more accurate.

Some Theory

Theorem (Lindenbaum, S, 2020)

Averaging over random subsets A of the equations of size $n < 0.9D$, we have

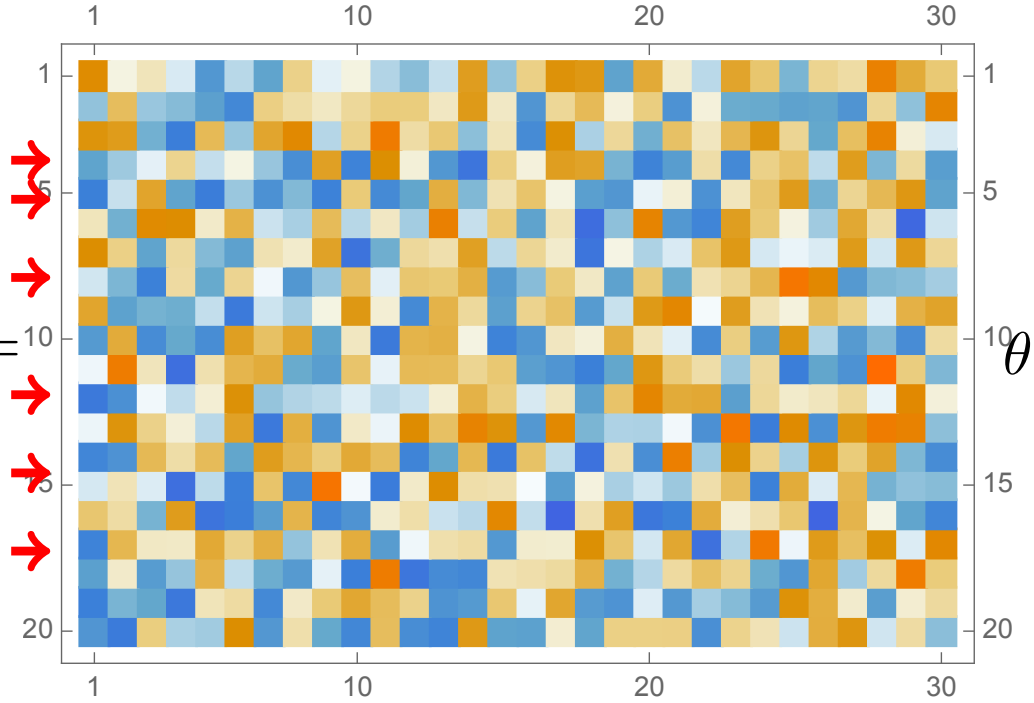
$$\mathbb{E}_{X,\omega} \left\| \mathbb{E}_A \left(\pi_A \theta^* - \widehat{\theta}_A \right) \right\| \lesssim \frac{n}{\sqrt{N}\sqrt{D}} + \frac{n}{D}.$$

This leads to a dichotomy:

1. If n is small, then the error is small but the projection $\pi_A \theta^*$ has little to do with the ground truth.
2. If n is large, then the error increases but the projection is more accurate.

Main Question

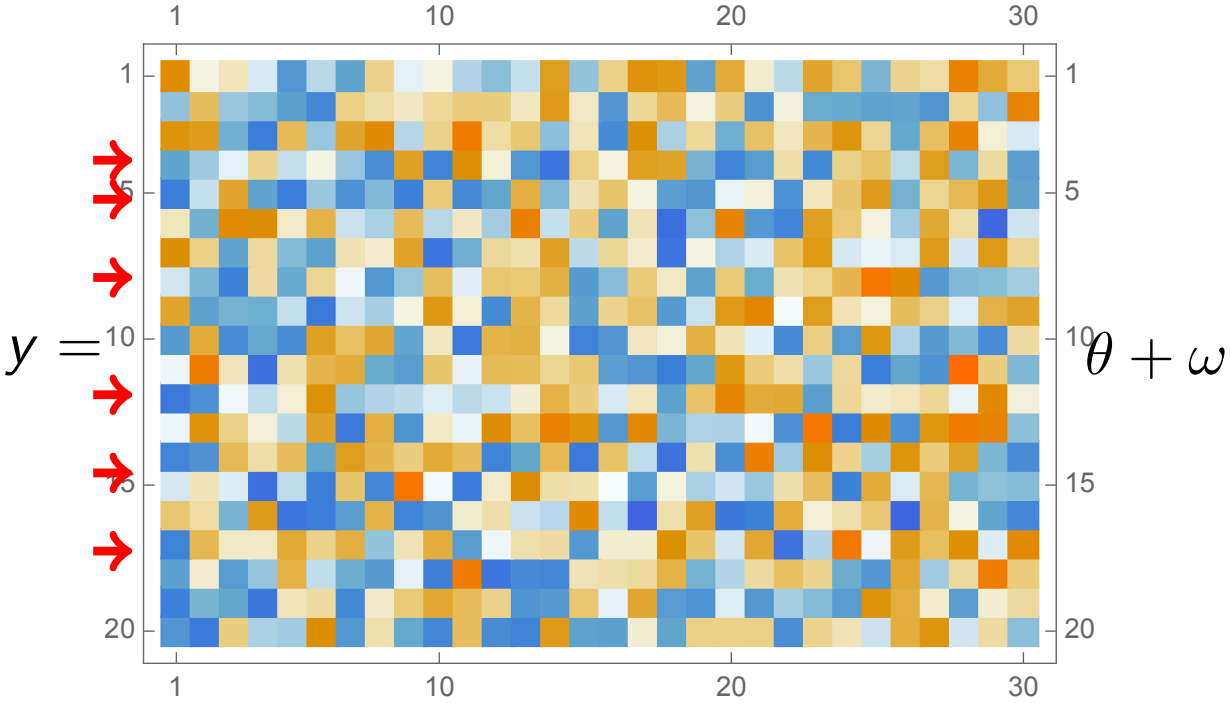
$y =$



$\theta + \omega$

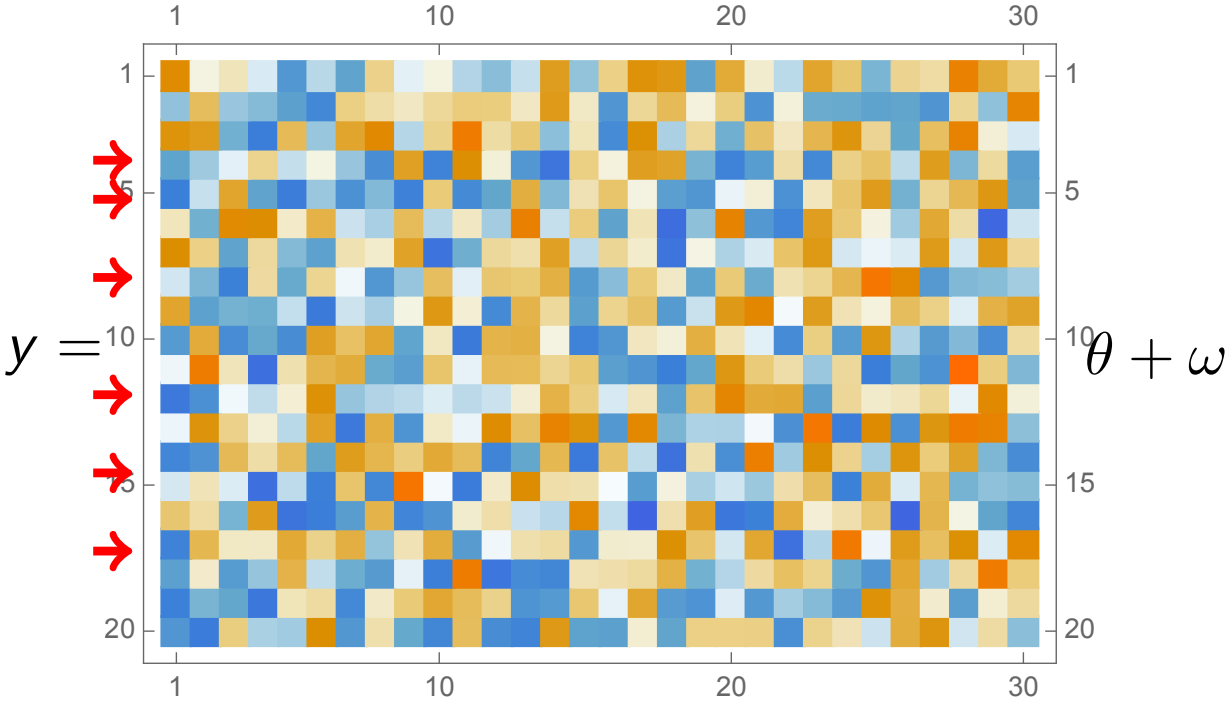
1. Which methods are best suited for this type of averaging?

Main Question

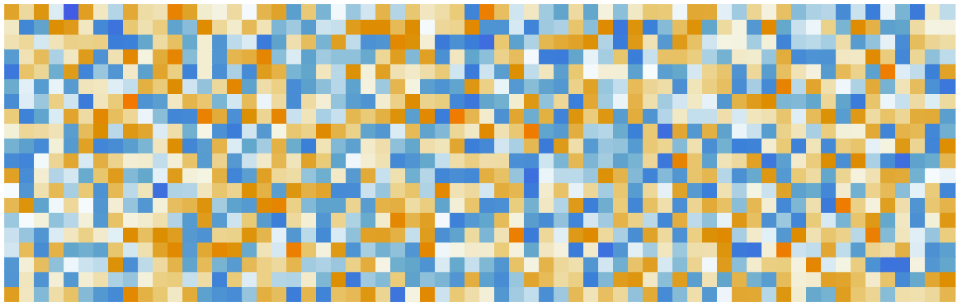


1. Which methods are best suited for this type of averaging?
2. Better way of selecting equations than just randomly?

Main Question



1. Which methods are best suited for this type of averaging?
2. Better way of selecting equations than just randomly?
3. Any chance of proving bounds with good constants?

$y =$  $\theta + \omega$

Thank you!