Adaptive Sparse Recovery

Eric Price

MIT

2012-04-26

Joint work with Piotr Indyk and David Woodruff, 2011-2012

Outline

Outline

Outline

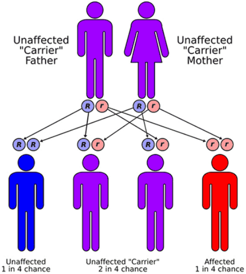
1. Motivating Example
2. Formal Introduction to Sparse Recovery/Compressive Sensing
3. Algorithm

Outline

Outline

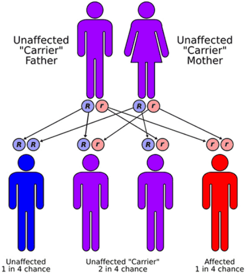
Carrier screening

- Want to figure out who carries a genetic mutation.



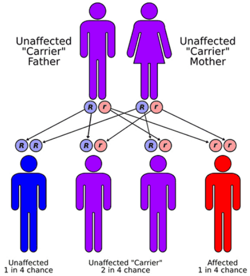
Carrier screening

- Want to figure out who carries a genetic mutation.



Carrier screening

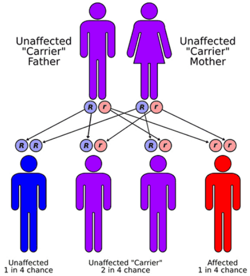
- Want to figure out who carries a genetic mutation.



- Test everyone!

Carrier screening

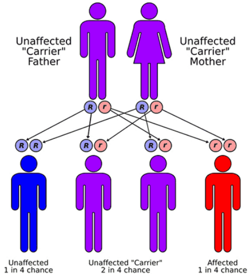
- Want to figure out who carries a genetic mutation.



- Test everyone!
- Test 1,000,000 people, find 1,000 carriers.

Carrier screening

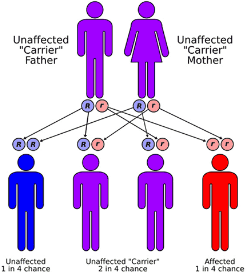
- Want to figure out who carries a genetic mutation.



- Test everyone!
- Test 1,000,000 people, find 1,000 carriers.
- Very inefficient.

Carrier screening

- Want to figure out who carries a genetic mutation.



- Test everyone!
- Test 1,000,000 people, find 1,000 carriers.
- Very inefficient.
- Idea: mix together samples.

Group testing

- Goal: find k carriers among n people.

Group testing

- Goal: find k carriers among n people.
- *Group testing*: test *groups* to see if any member is positive.

Group testing

- Goal: find k carriers among n people.
- *Group testing*: test *groups* to see if any member is positive.
- Doable with $\Theta(\log \binom{n}{k}) = \Theta(k \log(n/k))$ tests.

Group testing vs. compressive sensing

- Goal: find k carriers among n people.
- *Group testing*: test *groups* to see if any member is positive.
- Doable with $\Theta(\log \binom{n}{k}) = \Theta(k \log(n/k))$ tests.
- *Compressive sensing*: estimate the *number* of positives in group.

Group testing vs. compressive sensing

- Goal: find k carriers among n people.
- *Group testing*: test *groups* to see if any member is positive.
- Doable with $\Theta(\log \binom{n}{k}) = \Theta(k \log(n/k))$ tests.
- *Compressive sensing*: estimate the *number* of positives in group.
  - Trying to learn $x \in \mathbb{R}^n$. (Here, $x \in \{0, 1, 2\}^n$.)

Group testing vs. compressive sensing

- Goal: find k carriers among n people.
- *Group testing*: test *groups* to see if any member is positive.
- Doable with $\Theta(\log \binom{n}{k}) = \Theta(k \log(n/k))$ tests.
- *Compressive sensing*: estimate the *number* of positives in group.
  - Trying to learn $x \in \mathbb{R}^n$. (Here, $x \in \{0, 1, 2\}^n$.)
  - Choose coefficients $v \in \mathbb{R}^n$.

Group testing vs. compressive sensing

- Goal: find k carriers among n people.
- *Group testing*: test *groups* to see if any member is positive.
- Doable with $\Theta(\log \binom{n}{k}) = \Theta(k \log(n/k))$ tests.
- *Compressive sensing*: estimate the *number* of positives in group.
  - Trying to learn $x \in \mathbb{R}^n$. (Here, $x \in \{0, 1, 2\}^n$.)
  - Choose coefficients $v \in \mathbb{R}^n$.
  - Measure $\langle v, x \rangle$ with noise.

Group testing vs. compressive sensing

- Goal: find k carriers among n people.
- *Group testing*: test *groups* to see if any member is positive.
- Doable with $\Theta(\log \binom{n}{k}) = \Theta(k \log(n/k))$ tests.
- *Compressive sensing*: estimate the *number* of positives in group.
  - Trying to learn $x \in \mathbb{R}^n$. (Here, $x \in \{0, 1, 2\}^n$.)
  - Choose coefficients $v \in \mathbb{R}^n$.
  - Measure $\langle v, x \rangle$ with noise.
- Want to minimize number of tests.

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing		

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing	$\Theta(k \log(n/k))$	

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing	$\Theta(k \log(n/k))$	???

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing	$\Theta(k \log(n/k))$	???

- Expected fraction of DNA with mutation is $k/n = 0.1\%$.

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing	$\Theta(k \log(n/k))$	???

- Expected fraction of DNA with mutation is $k/n = 0.1\%$.
- Group testing possible: machine distinguishes 0% and 0.1%.

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing	$\Theta(k \log(n/k))$	???

- Expected fraction of DNA with mutation is $k/n = 0.1\%$.
- Group testing possible: machine distinguishes 0% and 0.1%.
- Also distinguishes 50.0% and 50.1%.

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing	$\Theta(k \log(n/k))$	???

- Expected fraction of DNA with mutation is $k/n = 0.1\%$.
- Group testing possible: machine distinguishes 0% and 0.1%.
- Also distinguishes 50.0% and 50.1%.
- Hope for $\log(n/k)$ bits/test, or $\Theta(k)$ measurements.

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing	$\Theta(k \log(n/k))$	???

- Expected fraction of DNA with mutation is $k/n = 0.1\%$.
- Group testing possible: machine distinguishes 0% and 0.1%.
- Also distinguishes 50.0% and 50.1%.
- Hope for $\log(n/k)$ bits/test, or $\Theta(k)$ measurements.
- Problem: any mixture has expected 0.1% mutation, so $O(1)$ bits.

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing	$\Theta(k \log(n/k))$	???

- Expected fraction of DNA with mutation is $k/n = 0.1\%$.
- Group testing possible: machine distinguishes 0% and 0.1%.
- Also distinguishes 50.0% and 50.1%.
- Hope for $\log(n/k)$ bits/test, or $\Theta(k)$ measurements.
- Problem: any mixture has expected 0.1% mutation, so $O(1)$ bits.
- Idea: use knowledge from early measurements to make later mixtures more concentrated with mutations.

Number of Measurements

	Non-adaptive	Adaptive
Group Testing	$\Omega(k^2)$	$\Theta(k \log(n/k))$
Compressive Sensing	$\Theta(k \log(n/k))$	$O(k \log\log(n/k))$

- Expected fraction of DNA with mutation is $k/n = 0.1\%$.
- Group testing possible: machine distinguishes 0% and 0.1%.
- Also distinguishes 50.0% and 50.1%.
- Hope for $\log(n/k)$ bits/test, or $\Theta(k)$ measurements.
- Problem: any mixture has expected 0.1% mutation, so $O(1)$ bits.
- Idea: use knowledge from early measurements to make later mixtures more concentrated with mutations.
- **This talk:** $O(k \log\log(n/k))$ adaptive linear measurements.

Outline

General Compressive Sensing

- Want to observe n-dimensional vector x
  - Which of n people have a genetic mutation.
  - Image
  - Traffic pattern of packets on a network.

General Compressive Sensing

- Want to observe n-dimensional vector x
  - Which of n people have a genetic mutation.
  - Image
  - Traffic pattern of packets on a network.
- Normally takes n observations to find.

General Compressive Sensing

- Want to observe n-dimensional vector x
  - Which of n people have a genetic mutation.
  - Image
  - Traffic pattern of packets on a network.
- Normally takes n observations to find.
- But we know some structure on the input:
  - Genetics: most people don't have the mutation.
  - Images: mostly smooth with some edges.
  - Traffic: Zipf distribution.

General Compressive Sensing

- Want to observe n-dimensional vector x
  - Which of n people have a genetic mutation.
  - Image
  - Traffic pattern of packets on a network.
- Normally takes n observations to find.
- But we know some structure on the input:
  - Genetics: most people don't have the mutation.
  - Images: mostly smooth with some edges.
  - Traffic: Zipf distribution.
- We use this structure to compress *space* (e.g. JPEG).

General Compressive Sensing

- Want to observe n-dimensional vector x
  - Which of n people have a genetic mutation.
  - Image
  - Traffic pattern of packets on a network.
- Normally takes n observations to find.
- But we know some structure on the input:
  - Genetics: most people don't have the mutation.
  - Images: mostly smooth with some edges.
  - Traffic: Zipf distribution.
- We use this structure to compress *space* (e.g. JPEG).
- Can we use structure to save on *observations*?

Cameras

- 5 megapixel camera takes 15 million byte-size observations.

Cameras

- 5 megapixel camera takes 15 million byte-size observations.
- Compresses it (JPEG) down to a million bytes.

Cameras

- 5 megapixel camera takes 15 million byte-size observations.
- Compresses it (JPEG) down to a million bytes.
- Why do we need to bother with so many observations? [Donoho,Candès-Tao]

Cameras

- 5 megapixel camera takes 15 million byte-size observations.
- Compresses it (JPEG) down to a million bytes.
- Why do we need to bother with so many observations? [Donoho,Candès-Tao]
- Cheap in visible light (silicon), very expensive in infrared.
  - $30,000 for 256x256 IR sensor.

Cameras

- 5 megapixel camera takes 15 million byte-size observations.
- Compresses it (JPEG) down to a million bytes.
- Why do we need to bother with so many observations? [Donoho,Candès-Tao]
- Cheap in visible light (silicon), very expensive in infrared.
  - $30,000 for 256x256 IR sensor.
- Use *structure* to take only a few million observations.

Cameras

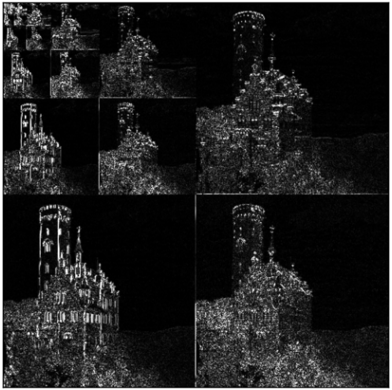
- 5 megapixel camera takes 15 million byte-size observations.
- Compresses it (JPEG) down to a million bytes.
- Why do we need to bother with so many observations? [Donoho,Candès-Tao]
- Cheap in visible light (silicon), very expensive in infrared.
  - $30,000 for 256x256 IR sensor.
- Use *structure* to take only a few million observations.
  - What structure? Sparsity.

Sparsity

- A vector is k-sparse if k components are non-zero.
- Images are almost sparse in the wavelet basis:

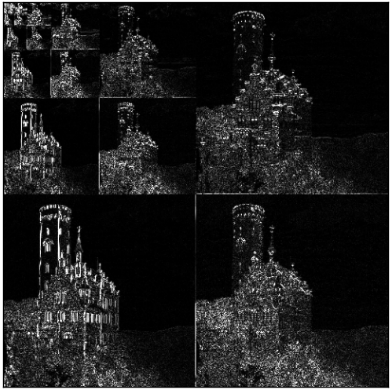
Sparsity

- A vector is *k*-sparse if *k* components are non-zero.
- Images are almost sparse in the wavelet basis:



Sparsity

- A vector is k-sparse if k components are non-zero.
- Images are almost sparse in the wavelet basis:



- Same kind of structure as in genetic testing!

Linear sketching/Compressive sensing

- Suppose an n-dimensional vector x is k-sparse in known basis.
- Given Ax, a set of $m << n$ linear products.

Linear sketching/Compressive sensing

- Suppose an n-dimensional vector x is k-sparse in known basis.
- Given Ax, a set of $m << n$ linear products.
- Why linear? Many applications:
  - Genetic testing: mixing blood samples.
  - Streaming updates: $A(x + \Delta) = Ax + A\Delta$.
  - Camera optics: filter in front of lens.

Linear sketching/Compressive sensing

- Suppose an n-dimensional vector x is k-sparse in known basis.
- Given Ax, a set of $m << n$ linear products.
- Why linear? Many applications:
  - Genetic testing: mixing blood samples.
  - Streaming updates: $A(x + \Delta) = Ax + A\Delta$.
  - Camera optics: filter in front of lens.
- Then it is possible to recover x from Ax.
  - Quickly
  - Robustly: get close to x if x is close to k-sparse.

Linear sketching/Compressive sensing

- Suppose an n-dimensional vector x is k-sparse in known basis.
- Given Ax, a set of $m << n$ linear products.
- Why linear? Many applications:
  - Genetic testing: mixing blood samples.
  - Streaming updates: $A(x + \Delta) = Ax + A\Delta$.
  - Camera optics: filter in front of lens.
- Then it is possible to recover x from Ax.
  - Quickly
  - Robustly: get close to x if x is close to k-sparse.
- Note: impossible without using sparsity (A is underdetermined).

Standard Sparse Recovery Framework

- Specify distribution on $m \times n$ matrices A (independent of x).

- Given linear sketch Ax, recover $\hat{x}$.
- Satisfying the recovery guarantee:

$$\|\hat{x} - x\|_2 \leqslant (1 + \epsilon) \min_{k\text{-sparse } x_k} \|x - x_k\|_2$$

with probability $2/3$.

Standard Sparse Recovery Framework

- Specify distribution on $m \times n$ matrices A (independent of x).

- Given linear sketch Ax, recover $\hat{x}$.
- Satisfying the recovery guarantee:

$$\|\hat{x} - x\|_2 \leqslant (1 + \epsilon) \min_{k\text{-sparse } x_k} \|x - x_k\|_2$$

  with probability $2/3$.
- Solvable with $O(\frac{1}{\epsilon} k \log \frac{n}{k})$ measurements [Candès-Romberg-Tao '06, Gilbert-Li-Porat-Strauss '10]

Standard Sparse Recovery Framework

- Specify distribution on $m \times n$ matrices A (independent of x).

- Given linear sketch Ax, recover $\hat{x}$.
- Satisfying the recovery guarantee:

$$\|\hat{x} - x\|_2 \leqslant (1 + \epsilon) \min_{k\text{-sparse } x_k} \|x - x_k\|_2$$

  with probability $2/3$.
- Solvable with $O(\frac{1}{\epsilon} k \log \frac{n}{k})$ measurements [Candès-Romberg-Tao '06, Gilbert-Li-Porat-Strauss '10]
- Matching lower bound. [Do Ba-Indyk-P-Woodruff '10, P-Woodruff '11]

Adaptive Sparse Recovery Framework

- For $i = 1 \ldots r$:
  - Choose matrix A_i based on previous observations (possibly randomized).
  - Observe $A_i x$.
  - Number of measurements m is total number of rows in all A_i.
  - Number of *rounds* is r.
- Given linear sketch Ax, recover $\hat{x}$.
- Satisfying the recovery guarantee:

$$\|\hat{x} - x\|_2 \leqslant (1 + \epsilon) \min_{k\text{-sparse } x_k} \|x - x_k\|_2$$

  with probability $2/3$.
- Solvable with $O(\frac{1}{\epsilon} k \log \frac{n}{k})$ measurements [Candès-Romberg-Tao '06, Gilbert-Li-Porat-Strauss '10]
- Matching lower bound. [Do Ba-Indyk-P-Woodruff '10, P-Woodruff '11]

Adaptive result in comparison to previous work

- Nonadaptive: $\Theta(\frac{1}{\epsilon}k \log \frac{n}{k})$.

Adaptive result in comparison to previous work

- Nonadaptive: $\Theta(\frac{1}{\epsilon} k \log \frac{n}{k})$.
- Adaptive: $O(k \log \frac{n}{k})$ with $\epsilon = o(1)$
  ([Haupt-Baraniuk-Castro-Nowak '09], in a slightly different setting)

Adaptive result in comparison to previous work

- Nonadaptive: $\Theta(\frac{1}{\epsilon} k \log \frac{n}{k})$.
- Adaptive: $O(k \log \frac{n}{k})$ with $\epsilon = o(1)$
  ([Haupt-Baraniuk-Castro-Nowak '09], in a slightly different setting)
- **This talk:** $O(\frac{1}{\epsilon} k \log \log \frac{n}{k})$. [Indyk-P-Woodruff '11]

Adaptive result in comparison to previous work

- Nonadaptive: $\Theta(\frac{1}{\epsilon} k \log \frac{n}{k})$.
- Adaptive: $O(k \log \frac{n}{k})$ with $\epsilon = o(1)$ ([Haupt-Baraniuk-Castro-Nowak '09], in a slightly different setting)
- **This talk:** $O(\frac{1}{\epsilon} k \log \log \frac{n}{k})$. [Indyk-P-Woodruff '11]
  - Using $r = O(\log \log n \log^* k)$ rounds.

Adaptive result in comparison to previous work

- Nonadaptive: $\Theta(\frac{1}{\epsilon} k \log \frac{n}{k})$.
- Adaptive: $O(k \log \frac{n}{k})$ with $\epsilon = o(1)$ ([Haupt-Baraniuk-Castro-Nowak '09], in a slightly different setting)
- **This talk:** $O(\frac{1}{\epsilon} k \log\log \frac{n}{k})$. [Indyk-P-Woodruff '11]
  - Using $r = O(\log\log n \log^* k)$ rounds.
- Even when $r = 2$, can get $O(k \log n + \frac{1}{\epsilon} k \log(k/\epsilon))$

Adaptive result in comparison to previous work

- Nonadaptive: $\Theta(\frac{1}{\epsilon} k \log \frac{n}{k})$.
- Adaptive: $O(k \log \frac{n}{k})$ with $\epsilon = o(1)$ ([Haupt-Baraniuk-Castro-Nowak '09], in a slightly different setting)
- **This talk:** $O(\frac{1}{\epsilon} k \log \log \frac{n}{k})$. [Indyk-P-Woodruff '11]
  - Using $r = O(\log \log n \log^* k)$ rounds.
- Even when $r = 2$, can get $O(k \log n + \frac{1}{\epsilon} k \log(k/\epsilon))$
  - Separating dependence on n and ϵ.

Applications of Adaptivity

- When does adaptivity make sense?

Applications of Adaptivity

- When does adaptivity make sense?
- Genetic testing:

Applications of Adaptivity

- When does adaptivity make sense?
- Genetic testing:
  - **Yes**, but multiple rounds can be costly.

Applications of Adaptivity

- When does adaptivity make sense?
- Genetic testing:
  - **Yes**, but multiple rounds can be costly.
- Cameras:

Applications of Adaptivity

- When does adaptivity make sense?
- Genetic testing:
  - **Yes**, but multiple rounds can be costly.
- Cameras:
  - Programmable pixels (mirrors or LCD display): **Yes**.

Applications of Adaptivity

- When does adaptivity make sense?
- Genetic testing:
  - **Yes**, but multiple rounds can be costly.
- Cameras:
  - Programmable pixels (mirrors or LCD display): **Yes.**
  - Hardwired lens: **No.**

Applications of Adaptivity

- When does adaptivity make sense?
- Genetic testing:
  - **Yes**, but multiple rounds can be costly.
- Cameras:
  - Programmable pixels (mirrors or LCD display): **Yes**.
  - Hardwired lens: **No**.
- Streaming algorithms:

Applications of Adaptivity

- When does adaptivity make sense?
- Genetic testing:
  - **Yes**, but multiple rounds can be costly.
- Cameras:
  - Programmable pixels (mirrors or LCD display): **Yes**.
  - Hardwired lens: **No**.
- Streaming algorithms:
  - Adaptivity corresponds to multiple passes.

Applications of Adaptivity

- When does adaptivity make sense?
- Genetic testing:
  - **Yes**, but multiple rounds can be costly.
- Cameras:
  - Programmable pixels (mirrors or LCD display): **Yes**.
  - Hardwired lens: **No**.
- Streaming algorithms:
  - Adaptivity corresponds to multiple passes.
  - Router finding most common connections: **No**.

Applications of Adaptivity

- When does adaptivity make sense?
- Genetic testing:
  - **Yes**, but multiple rounds can be costly.
- Cameras:
  - Programmable pixels (mirrors or LCD display): **Yes**.
  - Hardwired lens: **No**.
- Streaming algorithms:
  - Adaptivity corresponds to multiple passes.
  - Router finding most common connections: **No**.
  - Mapreduce finding most frequent URLs: **Yes**.

Outline

Outline of Algorithm

Theorem

Adaptive $1+\epsilon$*-approximate* k*-sparse recovery is possible with* $O(\frac{1}{\epsilon}k \log\log(n/k))$ *measurements.*

Outline of Algorithm

Theorem

Adaptive $1+\epsilon$*-approximate* k*-sparse recovery is possible with* $O(\frac{1}{\epsilon}k\log\log(n/k))$ *measurements.*

Lemma

Adaptive $O(1)$*-approximate* 1*-sparse recovery is possible with* $O(\log\log n)$ *measurements.*

Outline of Algorithm

Theorem

Adaptive $1 + \epsilon$*-approximate* k*-sparse recovery is possible with* $O(\frac{1}{\epsilon}k \log\log(n/k))$ *measurements.*

Lemma

Adaptive $O(1)$*-approximate* 1*-sparse recovery is possible with* $O(\log\log n)$ *measurements.*

- Lemma implies theorem using standard tricks ([GLPS10]).

1-sparse recovery: non-adaptive lower bound

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log\log n)$ measurements for some $C = O(1)$.

1-sparse recovery: non-adaptive lower bound

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log\log n)$ measurements for some $C = O(1)$.

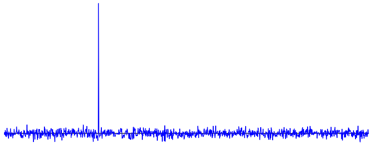
- Non-adaptive lower bound: why is this hard?

1-sparse recovery: non-adaptive lower bound

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log\log n)$ measurements for some $C = O(1)$.

- Non-adaptive lower bound: why is this hard?
- Hard case: x is random e_i plus Gaussian noise w.

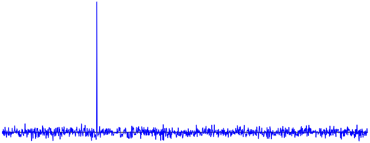


1-sparse recovery: non-adaptive lower bound

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log\log n)$ measurements for some $C = O(1)$.

- Non-adaptive lower bound: why is this hard?
- Hard case: x is random e_i plus Gaussian noise w.



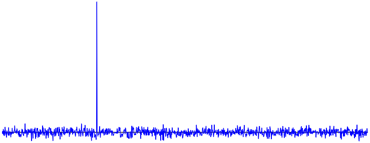
- Noise $\|w\|_2^2 = \Theta(1)$ so C-approximate recovery requires finding i.

1-sparse recovery: non-adaptive lower bound

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log\log n)$ measurements for some $C = O(1)$.

- Non-adaptive lower bound: why is this hard?
- Hard case: x is random e_i plus Gaussian noise w.



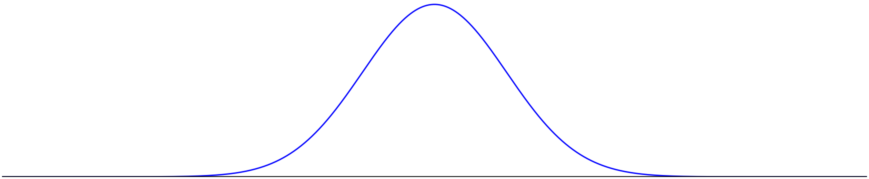
- Noise $\|w\|_2^2 = \Theta(1)$ so C-approximate recovery requires finding i.
- Observations $\langle v, x\rangle = v_i + \langle v, w\rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, for $z \sim N(0, \Theta(1))$.

1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x \rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, \Theta(1))$

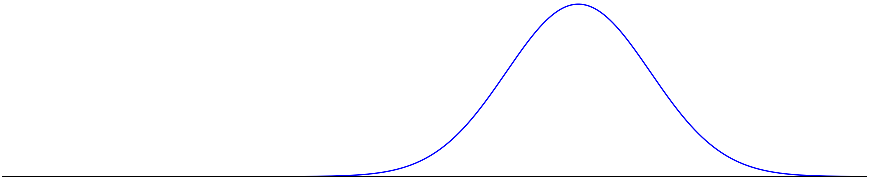
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x \rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, \Theta(1))$



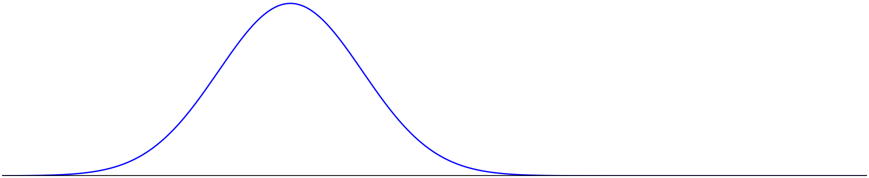
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x\rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, \Theta(1))$



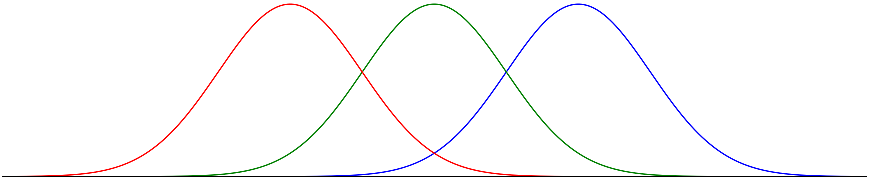
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x\rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, \Theta(1))$



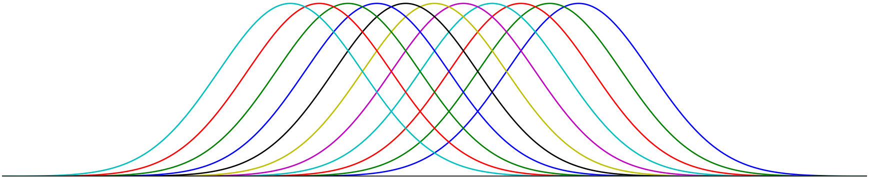
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x \rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, \Theta(1))$



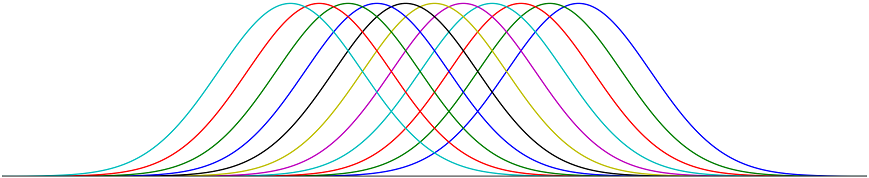
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x\rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, \Theta(1))$



1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x\rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, \Theta(1))$



- Information capacity

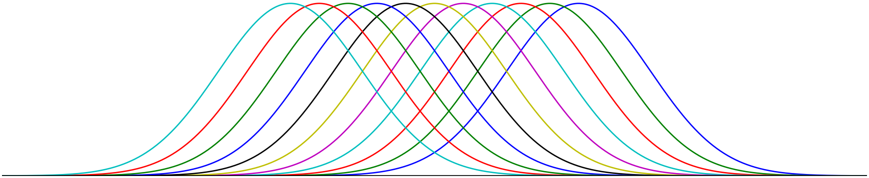
$$I(i, \langle v, x\rangle) \leqslant \frac{1}{2}\log(1 + \mathrm{SNR})$$

where *SNR* denotes the "signal-to-noise ratio,"

$$SNR = \frac{\mathbb{E}[\text{signal}^2]}{\mathbb{E}[\text{noise}^2]} \lesssim \frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n} = 1$$

1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x\rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, \Theta(1))$



- Information capacity

$$I(i, \langle v, x\rangle) \leqslant \frac{1}{2}\log(1 + \mathrm{SNR})$$

where *SNR* denotes the "signal-to-noise ratio,"

$$SNR = \frac{\mathbb{E}[\text{signal}^2]}{\mathbb{E}[\text{noise}^2]} \lesssim \frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n} = 1$$

- Finding i needs $\Omega(\log n)$ non-adaptive measurements.

1-sparse recovery: changes in adaptive setting

- Information capacity

$$I(i, \langle v, x \rangle) \leqslant \frac{1}{2}\log(1 + \mathrm{SNR}).$$

where *SNR* denotes the "signal-to-noise ratio,"

$$SNR = \Theta\left(\frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n}\right).$$

1-sparse recovery: changes in adaptive setting

- Information capacity

$$I(i, \langle v, x \rangle) \leqslant \frac{1}{2} \log(1 + \text{SNR}).$$

where *SNR* denotes the "signal-to-noise ratio,"

$$SNR = \Theta\left(\frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n}\right).$$

- If i is independent of v, this is $O(1)$.

1-sparse recovery: changes in adaptive setting

- Information capacity

$$I(i, \langle v, x \rangle) \leqslant \frac{1}{2} \log(1 + \mathrm{SNR}).$$

where *SNR* denotes the "signal-to-noise ratio,"

$$SNR = \Theta\left(\frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n}\right).$$

- If i is independent of v, this is $O(1)$.
- As we learn about i, we can increase $\mathbb{E}[v_i^2]$ for constant $\|v\|_2$.

1-sparse recovery: changes in adaptive setting

- Information capacity

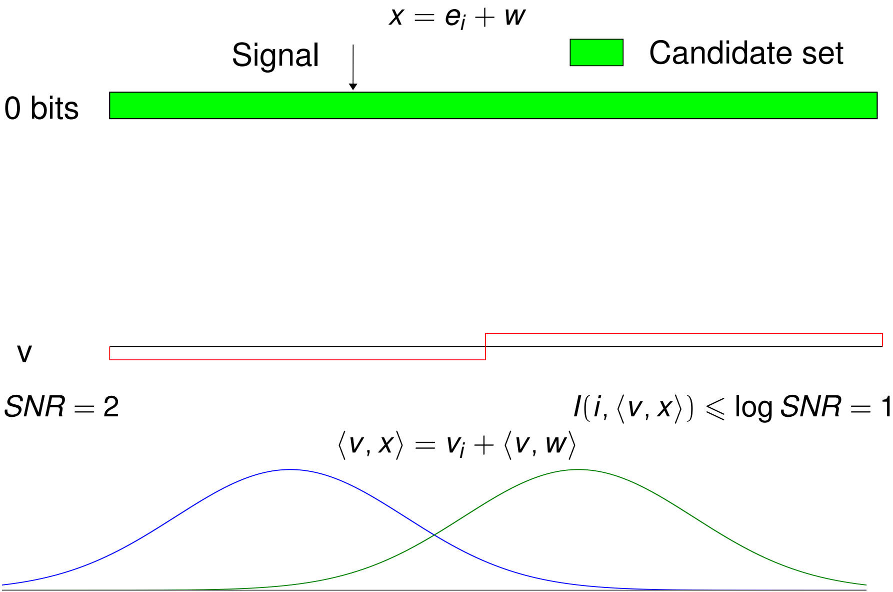
$$I(i, \langle v, x \rangle) \leqslant \frac{1}{2} \log(1 + \mathrm{SNR}).$$

where *SNR* denotes the "signal-to-noise ratio,"

$$SNR = \Theta\left(\frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n}\right).$$

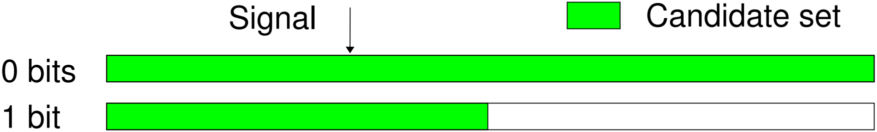
- If i is independent of v, this is $O(1)$.
- As we learn about i, we can increase $\mathbb{E}[v_i^2]$ for constant $\|v\|_2$.
  - Equivalently, for constant $\mathbb{E}[v_i^2]$ we can decrease $\|v\|_2$.

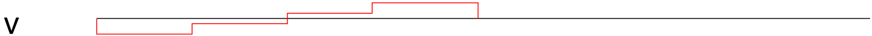
1-sparse recovery: idea



1-sparse recovery: idea

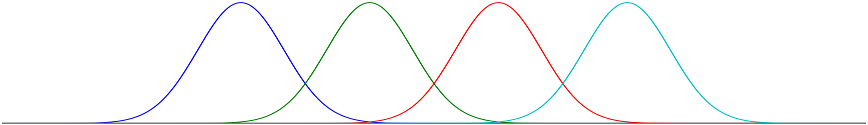
$$x = e_i + w$$





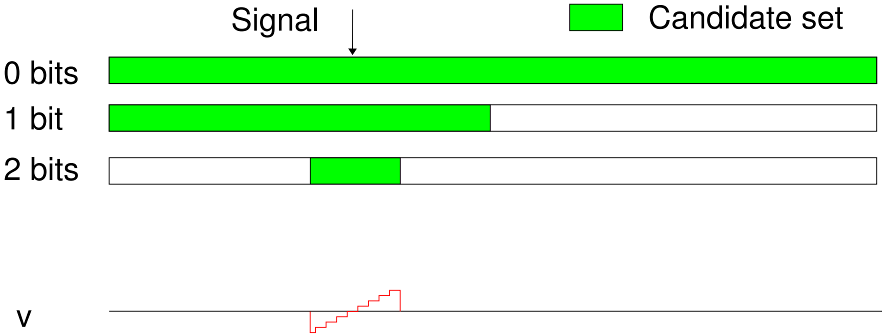
$SNR = 2^2$ $\qquad$ $I(i, \langle v, x \rangle) \leqslant \log SNR = 2$

$$\langle v, x \rangle = v_i + \langle v, w \rangle$$



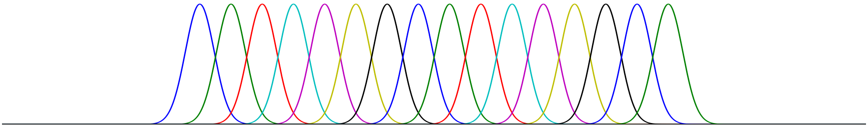
1-sparse recovery: idea

$$x = e_i + w$$



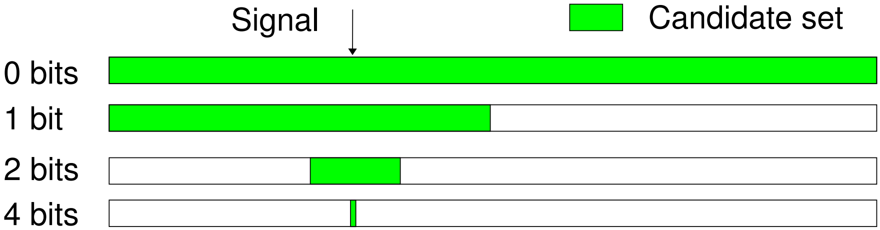
$SNR = 2^4$ $\qquad$ $I(i, \langle v, x\rangle) \leqslant \log SNR = 4$

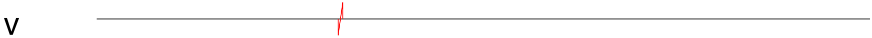
$$\langle v, x\rangle = v_i + \langle v, w\rangle$$



1-sparse recovery: idea

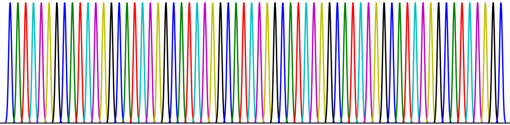
$$x = e_i + w$$





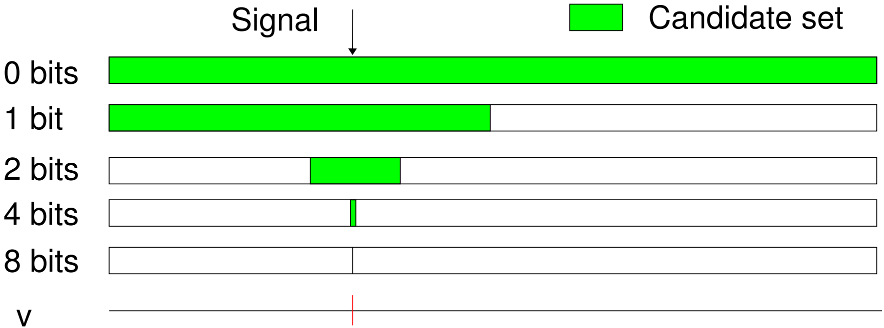
$SNR = 2^8$ $\qquad$ $I(i, \langle v, x \rangle) \leqslant \log SNR = 8$

$$\langle v, x \rangle = v_i + \langle v, w \rangle$$



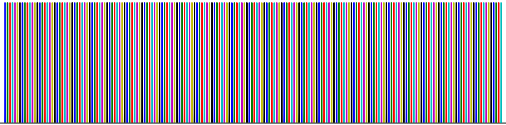
1-sparse recovery: idea

$$x = e_i + w$$



$SNR = 2^{16}$ $\qquad I(i, \langle v, x \rangle) \leqslant \log SNR = 16$

$$\langle v, x \rangle = v_i + \langle v, w \rangle$$



Goal

- Shown intuition for specific distribution on x

Goal

- Shown intuition for specific distribution on x
- Match previous convergence for arbitrary $x = \alpha e_i + w$?

Goal

- Shown intuition for specific distribution on x
- Match previous convergence for arbitrary $x = \alpha e_i + w$?
  - α may not be 1.

Goal

- Shown intuition for specific distribution on x
- Match previous convergence for arbitrary $x = \alpha e_i + w$?
  - α may not be 1.
  - Work for a *specific* x with $3/4$ probability.

Goal

- Shown intuition for specific distribution on x
- Match previous convergence for arbitrary $x = \alpha e_i + w$?
  - α may not be 1.
  - Work for a *specific* x with $3/4$ probability.
  - Distribution over A, for fixed w.

Goal

- Find i from $x = \alpha e_i + w$ using $\log\log n$ adaptive measurements.

Goal

- Find i from $x = \alpha e_i + w$ using $\log\log n$ adaptive measurements.
- Define the signal-to-noise ratio

$$SNR(x) = \alpha^2 / \|w\|_2^2.$$

Goal

- Find i from $x = \alpha e_i + w$ using $\log \log n$ adaptive measurements.
- Define the signal-to-noise ratio

$$SNR(x) = \alpha^2 / \|w\|_2^2.$$

For Gaussian w, can fit roughly $\sqrt{SNR}$ distinct Gaussians.

Goal

- Find i from $x = \alpha e_i + w$ using $\log \log n$ adaptive measurements.
- Define the signal-to-noise ratio

$$SNR(x) = \alpha^2 / \|w\|_2^2.$$

For Gaussian w, can fit roughly $\sqrt{SNR}$ distinct Gaussians.

- Given $O(1)$ measurements, find $S \ni i$ with

$$SNR(x_S) \geqslant \quad (SNR(x))^{3/2}$$

Goal

- Find i from $x = \alpha e_i + w$ using $\log \log n$ adaptive measurements.
- Define the signal-to-noise ratio

$$SNR(x) = \alpha^2 / \|w\|_2^2.$$

For Gaussian w, can fit roughly $\sqrt{SNR}$ distinct Gaussians.

- Given $O(1)$ measurements, find $S \ni i$ with

$$SNR(x_S) \geqslant \delta^2 (SNR(x))^{3/2}$$

with probability $1 - O(\delta)$.

Goal

- Find i from $x = \alpha e_i + w$ using $\log\log n$ adaptive measurements.
- Define the signal-to-noise ratio

$$SNR(x) = \alpha^2 / \|w\|_2^2.$$

For Gaussian w, can fit roughly $\sqrt{SNR}$ distinct Gaussians.
- Given $O(1)$ measurements, find $S \ni i$ with

$$SNR(x_S) \geqslant \delta^2 (SNR(x))^{3/2}$$

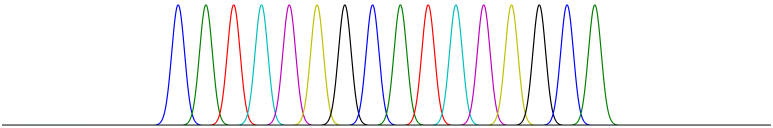
with probability $1 - O(\delta)$.
- Repeat on x_S.

Goal

- Find i from $x = \alpha e_i + w$ using $\log\log n$ adaptive measurements.
- Define the signal-to-noise ratio
$$SNR(x) = \alpha^2/\|w\|_2^2.$$
For Gaussian w, can fit roughly $\sqrt{SNR}$ distinct Gaussians.
- Given $O(1)$ measurements, find $S \ni i$ with
$$SNR(x_S) \geqslant \delta^2(SNR(x))^{3/2}$$
with probability $1 - O(\delta)$.
- Repeat on x_S.
- Once $SNR(x)$ reaches $O(n^2)$, will have $S = \{i\}$.

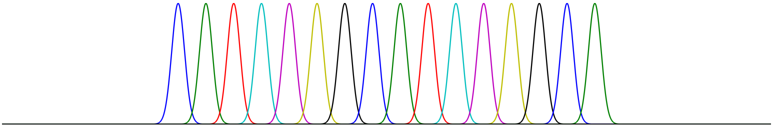
Recovery when $SNR > n^2$

Getting $\log n$ bits when SNR is n^2



Recovery when $SNR > n^2$

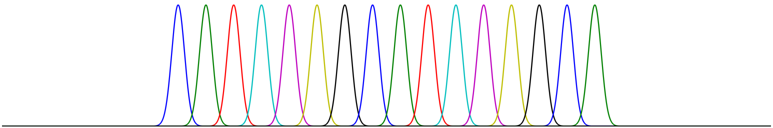
Getting $\log n$ bits when SNR is n^2



- Find i in 2 measurements with probability $1 - O(\delta)$.

Recovery when $SNR > n^2$

Getting $\log n$ bits when SNR is n^2

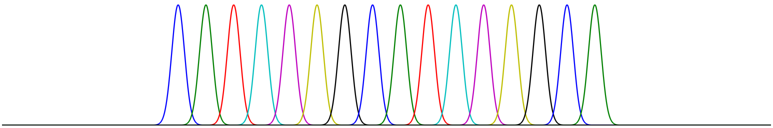


- Find i in 2 measurements with probability $1 - O(\delta)$.
- Observe

$$a = \sum jx_j \qquad b = \sum x_j$$

Recovery when $SNR > n^2$

Getting $\log n$ bits when SNR is n^2



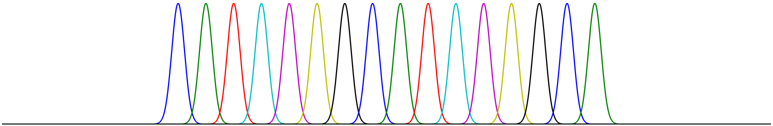
- Find i in 2 measurements with probability $1 - O(\delta)$.
- Observe

$$a = \sum jx_j \qquad\qquad b = \sum x_j$$

- Then $i \approx a/b$

Recovery when $SNR > n^2$

Getting $\log n$ bits when SNR is n^2



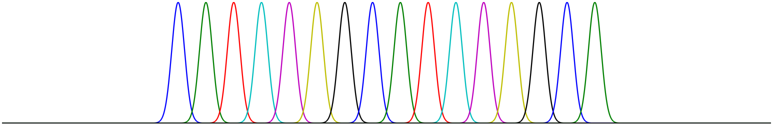
- Find i in 2 measurements with probability $1 - O(\delta)$.
- Observe

$$a = \sum jx_j \qquad\qquad b = \sum x_j$$

- Then $i \approx a/b$, with error proportional to $\|w\|_1$.

Recovery when $SNR > n^2$

Getting $\log n$ bits when SNR is n^2



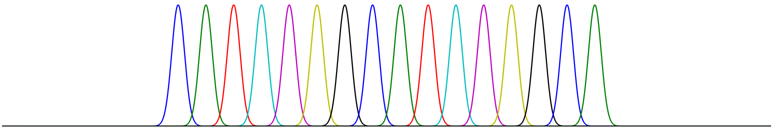
- Find i in 2 measurements with probability $1 - O(\delta)$.
- Observe, for $s \in \{\pm 1\}^n$ pairwise independently:

$$a = \sum j x_j s_j \qquad\qquad b = \sum x_j s_j$$

- Then $i \approx a/b$, with error proportional to $\|w\|_2$.

Recovery when $SNR > n^2$

Getting $\log n$ bits when SNR is n^2



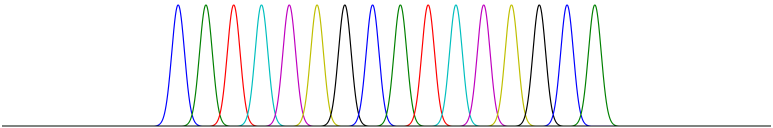
- Find i in 2 measurements with probability $1 - O(\delta)$.
- Observe, for $s \in \{\pm 1\}^n$ pairwise independently:

$$a = \sum j x_j s_j \qquad\qquad b = \sum x_j s_j$$

- Then $i \approx a/b$, with error proportional to $\|w\|_2$.
- $|i - a/b| < n/(\delta\sqrt{SNR})$ with probability $1 - O(\delta)$ (over s).

Recovery when $SNR > n^2$

Getting $\log n$ bits when SNR is n^2

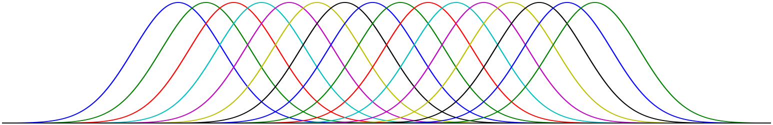


- Find i in 2 measurements with probability $1 - O(\delta)$.
- Observe, for $s \in \{\pm 1\}^n$ pairwise independently:

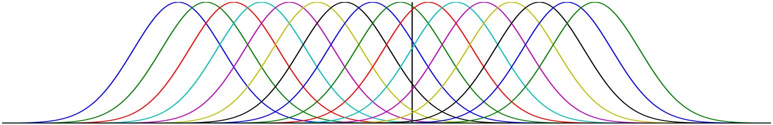
$$a = \sum j x_j s_j \qquad\qquad b = \sum x_j s_j$$

- Then $i \approx a/b$, with error proportional to $\|w\|_2$.
- $|i - a/b| < n/(\delta\sqrt{SNR})$ with probability $1 - O(\delta)$ (over s).
- For $SNR > (n/\delta)^2$, done.

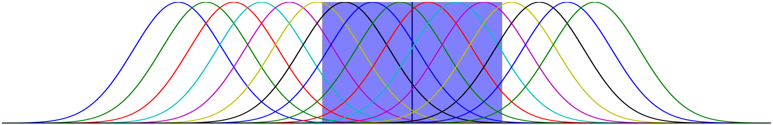
Getting log SNR bits for general SNR



Getting $\log SNR$ bits for general SNR

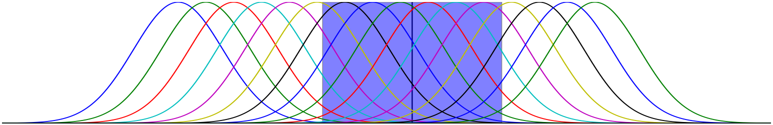


Getting log *SNR* bits for general *SNR*



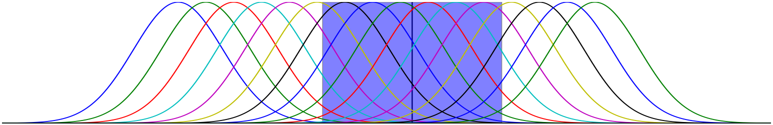
- Still $|i - a/b| < n/(2\delta\sqrt{SNR})$ with $1 - O(\delta)$ probability.

Getting log *SNR* bits for general *SNR*



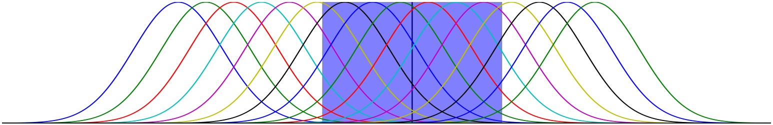
- Still $|i - a/b| < n/(2\delta\sqrt{SNR})$ with $1 - O(\delta)$ probability.
- So given a and b, know i in S of size $|S| = n/(\delta\sqrt{SNR})$

Getting log *SNR* bits for general *SNR*

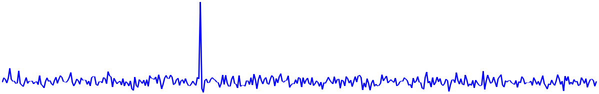


- Still $|i - a/b| < n/(2\delta\sqrt{SNR})$ with $1 - O(\delta)$ probability.
- So given a and b, know i in S of size $|S| = n/(\delta\sqrt{SNR})$
- Want $SNR(x_S) \approx (\delta\sqrt{SNR(x)})SNR(x)$.

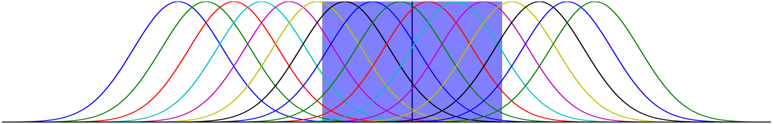
Getting $\log SNR$ bits for general SNR



- Still $|i - a/b| < n/(2\delta\sqrt{SNR})$ with $1 - O(\delta)$ probability.
- So given a and b, know i in S of size $|S| = n/(\delta\sqrt{SNR})$
- Want $SNR(x_S) \approx (\delta\sqrt{SNR(x)})SNR(x)$.

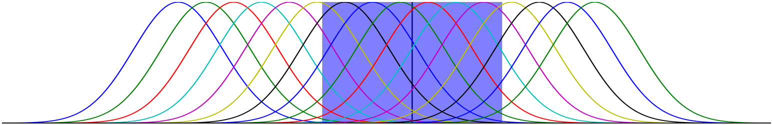


Getting $\log SNR$ bits for general SNR

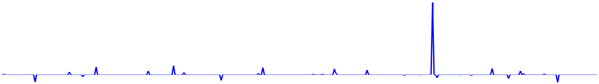


- Still $|i - a/b| < n/(2\delta\sqrt{SNR})$ with $1 - O(\delta)$ probability.
- So given a and b, know i in S of size $|S| = n/(\delta\sqrt{SNR})$
- Want $SNR(x_S) \approx (\delta\sqrt{SNR(x)})SNR(x)$.

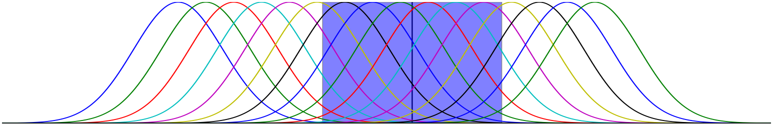
Getting log *SNR* bits for general *SNR*



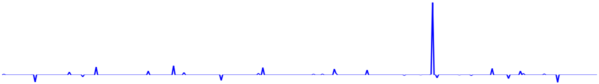
- Still $|i - a/b| < n/(2\delta\sqrt{SNR})$ with $1 - O(\delta)$ probability.
- So given a and b, know i in S of size $|S| = n/(\delta\sqrt{SNR})$
- Want $SNR(x_S) \approx (\delta\sqrt{SNR(x)})SNR(x)$.



Getting log SNR bits for general SNR



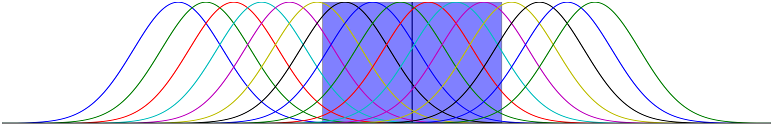
- Still $|i - a/b| < n/(2\delta\sqrt{SNR})$ with $1 - O(\delta)$ probability.
- So given a and b, know i in S of size $|S| = n/(\delta\sqrt{SNR})$
- Want $SNR(x_S) \approx (\delta\sqrt{SNR(x)})SNR(x)$.



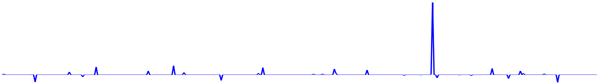
- Randomly permute x beforehand! Then SNR shrinks in expectation.

$$SNR(x_S) \geqslant (\delta SNR(x))^{3/2}$$

Getting log *SNR* bits for general *SNR*



- Still $|i - a/b| < n/(2\delta\sqrt{SNR})$ with $1 - O(\delta)$ probability.
- So given a and b, know i in S of size $|S| = n/(\delta\sqrt{SNR})$
- Want $SNR(x_S) \approx (\delta\sqrt{SNR(x)})SNR(x)$.



- Randomly permute x beforehand! Then SNR shrinks in expectation.

$$SNR(x_S) \geqslant (\delta SNR(x))^{3/2}$$

- Set $\delta = 0.1/2^r$ in round r; still doubly exponential growth.

End proof of key lemma

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log\log n)$ measurements for some $C = O(1)$.

End proof of key lemma

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log \log n)$ measurements for some $C = O(1)$.

Theorem (Adaptive upper bound)

Adaptive $1 + \epsilon$-approximate k-sparse recovery is possible with $O(\frac{1}{\epsilon} k \log \log(n/k))$ measurements.

End proof of key lemma

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log \log n)$ measurements for some $C = O(1)$.

Theorem (Adaptive upper bound)

Adaptive $1 + \epsilon$-approximate k-sparse recovery is possible with $O(\frac{1}{\epsilon} k \log \log(n/k))$ measurements.

- Lemma implies theorem using standard tricks (a la [GLPS10]):

End proof of key lemma

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log\log n)$ measurements for some $C = O(1)$.

Theorem (Adaptive upper bound)

Adaptive $1+\epsilon$-approximate k-sparse recovery is possible with $O(\frac{1}{\epsilon}k\log\log(n/k))$ measurements.

- Lemma implies theorem using standard tricks (a la [GLPS10]):
  - Subsample at rate ϵ/k and apply the lemma, $O(k/\epsilon)$ times.

End proof of key lemma

Lemma

Adaptive C-approximate 1-sparse recovery is possible with $O(\log\log n)$ measurements for some $C = O(1)$.

Theorem (Adaptive upper bound)

Adaptive $1+\epsilon$-approximate k-sparse recovery is possible with $O(\frac{1}{\epsilon}k\log\log(n/k))$ measurements.

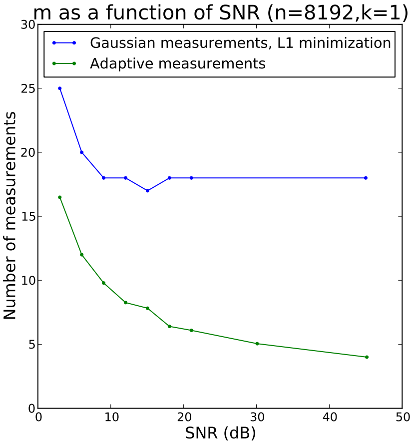
- Lemma implies theorem using standard tricks (a la [GLPS10]):
  - Subsample at rate ϵ/k and apply the lemma, $O(k/\epsilon)$ times.
  - Replace k by $k/2$, repeat.

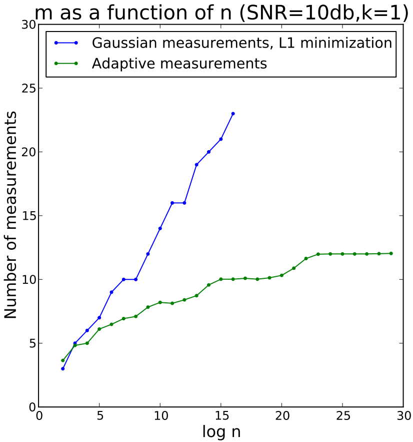
Experiments!

Does $O(\log n) \to O(\log \log n)$ really matter? What about the constants?

Experiments!

Does $O(\log n) \to O(\log\log n)$ really matter? What about the constants?





Round complexity

- Basic algorithm
  - $O(\frac{1}{\epsilon} k \log\log(n/k))$ measurements.
  - $O(\log^* k \log\log n)$ rounds.

Round complexity

- Basic algorithm
  - $O(\frac{1}{\epsilon}k \log\log(n/k))$ measurements.
  - $O(\log^* k \log\log n)$ rounds.
- Given $O(r \log^* k)$ rounds, $O(\frac{1}{\epsilon}kr \log^{1/r}(n/k))$ measurements.

Round complexity

- Basic algorithm
  - $O(\frac{1}{\epsilon}k \log\log(n/k))$ measurements.
  - $O(\log^* k \log\log n)$ rounds.
- Given $O(r \log^* k)$ rounds, $O(\frac{1}{\epsilon}kr \log^{1/r}(n/k))$ measurements.
- Lower bound: given r rounds, $\Omega(k/\epsilon + r \log^{1/r} n)$ measurements. [Arias-Castro-Càndes-Davenport '11, P-Woodruff '12]

Round complexity

- Basic algorithm
  - $O(\frac{1}{\epsilon} k \log\log(n/k))$ measurements.
  - $O(\log^* k \log\log n)$ rounds.
- Given $O(r \log^* k)$ rounds, $O(\frac{1}{\epsilon} kr \log^{1/r}(n/k))$ measurements.
- Lower bound: given r rounds, $\Omega(k/\epsilon + r \log^{1/r} n)$ measurements. [Arias-Castro-Càndes-Davenport '11, P-Woodruff '12]
  - For $k = 1$, tight up to $O(\log^* k)$ factor in rounds.

Round complexity

- Basic algorithm
  - $O(\frac{1}{\epsilon} k \log \log(n/k))$ measurements.
  - $O(\log^* k \log \log n)$ rounds.
- Given $O(r \log^* k)$ rounds, $O(\frac{1}{\epsilon} kr \log^{1/r}(n/k))$ measurements.
- Lower bound: given r rounds, $\Omega(k/\epsilon + r \log^{1/r} n)$ measurements. [Arias-Castro-Càndes-Davenport '11, P-Woodruff '12]
  - For $k = 1$, tight up to $O(\log^* k)$ factor in rounds.
- Given two rounds, $O(\frac{1}{\epsilon} k \log(k/\epsilon) + k \log(n/k))$ measurements.

Round complexity

- Basic algorithm
  - $O(\frac{1}{\epsilon}k \log\log(n/k))$ measurements.
  - $O(\log^* k \log\log n)$ rounds.
- Given $O(r \log^* k)$ rounds, $O(\frac{1}{\epsilon}kr \log^{1/r}(n/k))$ measurements.
- Lower bound: given r rounds, $\Omega(k/\epsilon + r \log^{1/r} n)$ measurements. [Arias-Castro-Càndes-Davenport '11, P-Woodruff '12]
  - For $k = 1$, tight up to $O(\log^* k)$ factor in rounds.
- Given two rounds, $O(\frac{1}{\epsilon}k \log(k/\epsilon) + k \log(n/k))$ measurements.
  - Separates dependence on ϵ and n.

Outline

Results and future work

- Nonadaptive sparse recovery requires $\Theta(k \log \frac{n}{k})$ measurements.

Results and future work

- Nonadaptive sparse recovery requires $\Theta(k \log \frac{n}{k})$ measurements.
- Adaptive algorithm uses $O(r \log^* k)$ rounds for $O(\frac{1}{\epsilon} kr \log^{1/r} \frac{n}{k})$ measurements.

Results and future work

- Nonadaptive sparse recovery requires $\Theta(k \log \frac{n}{k})$ measurements.
- Adaptive algorithm uses $O(r \log^* k)$ rounds for $O(\frac{1}{\epsilon} kr \log^{1/r} \frac{n}{k})$ measurements.
  - Also: 2 rounds, $O(\frac{1}{\epsilon} k \log(k/\epsilon) + k \log(n/k))$ measurements.

Results and future work

- Nonadaptive sparse recovery requires $\Theta(k \log \frac{n}{k})$ measurements.
- Adaptive algorithm uses $O(r \log^* k)$ rounds for $O(\frac{1}{\epsilon} kr \log^{1/r} \frac{n}{k})$ measurements.
  - Also: 2 rounds, $O(\frac{1}{\epsilon} k \log(k/\epsilon) + k \log(n/k))$ measurements.
- Clearer characterization of measurement/round tradeoff?
  - Algorithm is $O(\log^* k)$ rounds off lower bound.
  - Given 4 iterations, how many total blood tests do we need?

Results and future work

- Nonadaptive sparse recovery requires $\Theta(k \log \frac{n}{k})$ measurements.
- Adaptive algorithm uses $O(r \log^* k)$ rounds for $O(\frac{1}{\epsilon} kr \log^{1/r} \frac{n}{k})$ measurements.
  - Also: 2 rounds, $O(\frac{1}{\epsilon} k \log(k/\epsilon) + k \log(n/k))$ measurements.
- Clearer characterization of measurement/round tradeoff?
  - Algorithm is $O(\log^* k)$ rounds off lower bound.
  - Given 4 iterations, how many total blood tests do we need?
- Incorporating adaptivity in constrained matrix designs?

Results and future work

- Nonadaptive sparse recovery requires $\Theta(k \log \frac{n}{k})$ measurements.
- Adaptive algorithm uses $O(r \log^* k)$ rounds for $O(\frac{1}{\epsilon} kr \log^{1/r} \frac{n}{k})$ measurements.
  - Also: 2 rounds, $O(\frac{1}{\epsilon} k \log(k/\epsilon) + k \log(n/k))$ measurements.
- Clearer characterization of measurement/round tradeoff?
  - Algorithm is $O(\log^* k)$ rounds off lower bound.
  - Given 4 iterations, how many total blood tests do we need?
- Incorporating adaptivity in constrained matrix designs?
- Relate k/ϵ and n in tight bounds? Know $\Omega(\frac{1}{\epsilon} k + r \log^{1/r} n)$.

Separating ϵ and n

- Hash to $O(k^2/\epsilon^2)$ blocks, and probably all of:

Separating ϵ and n

- Hash to $O(k^2/\epsilon^2)$ blocks, and probably all of:
  - A perfect hash, so heavy hitters land in different blocks.

Separating ϵ and n

- Hash to $O(k^2/\epsilon^2)$ blocks, and probably all of:
  - A perfect hash, so heavy hitters land in different blocks.
  - Each heavy hitter dominates the noise in the same block.

Separating ϵ and n

- Hash to $O(k^2/\epsilon^2)$ blocks, and probably all of:
  - A perfect hash, so heavy hitters land in different blocks.
  - Each heavy hitter dominates the noise in the same block.
  - Overall, the noise grows by at most $1 + \epsilon/2$ factor

Separating ϵ and n

- Hash to $O(k^2/\epsilon^2)$ blocks, and probably all of:
  - A perfect hash, so heavy hitters land in different blocks.
  - Each heavy hitter dominates the noise in the same block.
  - Overall, the noise grows by at most $1 + \epsilon/2$ factor
- Solve $(1 + \epsilon)$-approximate sparse recovery in reduced space: $O(\frac{1}{\epsilon}k\log(k/\epsilon))$

Separating ϵ and n

- Hash to $O(k^2/\epsilon^2)$ blocks, and probably all of:
  - A perfect hash, so heavy hitters land in different blocks.
  - Each heavy hitter dominates the noise in the same block.
  - Overall, the noise grows by at most $1 + \epsilon/2$ factor
- Solve $(1 + \epsilon)$-approximate sparse recovery in reduced space: $O(\frac{1}{\epsilon} k \log(k/\epsilon))$
- Identifies $O(k)$ blocks to search containing enough heavy hitter mass.

Separating ϵ and n

- Hash to $O(k^2/\epsilon^2)$ blocks, and probably all of:
  - A perfect hash, so heavy hitters land in different blocks.
  - Each heavy hitter dominates the noise in the same block.
  - Overall, the noise grows by at most $1 + \epsilon/2$ factor
- Solve $(1 + \epsilon)$-approximate sparse recovery in reduced space: $O(\frac{1}{\epsilon} k \log(k/\epsilon))$
- Identifies $O(k)$ blocks to search containing enough heavy hitter mass.
- Heavy hitters are $O(1)$-heavy among their blocks, so $O(\log n)$ per block suffices.

Separating ϵ and n

- Hash to $O(k^2/\epsilon^2)$ blocks, and probably all of:
  - A perfect hash, so heavy hitters land in different blocks.
  - Each heavy hitter dominates the noise in the same block.
  - Overall, the noise grows by at most $1 + \epsilon/2$ factor
- Solve $(1 + \epsilon)$-approximate sparse recovery in reduced space: $O(\frac{1}{\epsilon} k \log(k/\epsilon))$
- Identifies $O(k)$ blocks to search containing enough heavy hitter mass.
- Heavy hitters are $O(1)$-heavy among their blocks, so $O(\log n)$ per block suffices.
- Result: $O(\frac{1}{\epsilon} k \log(k/\epsilon) + k \log n)$.