COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco
University of Massachusetts Amherst. Fall 2019.
Lecture 23
• Problem Set 4 due Sunday 12/15 at 8pm.
• Exam prep materials posted under the ‘Schedule’ tab of the course page.
• SRTI survey is open until 12/22. Your feedback this semester has been very helpful to me, so please fill out the survey!
• https://owl.umass.edu/partners/courseEvalSurvey/uma/
Last Class:

• Some counterintuitive properties of high dimensional space.
• Connections to ‘curse of dimensionality’.
Last Class:

- Some counterintuitive properties of high dimensional space.
- Connections to ‘curse of dimensionality’.

This Class:

- Compressed sensing and sparse recovery.
- Applications to sparse regression, frequent elements problem, sparse Fourier transform, efficient imaging, etc.
Consider matrix $A \in \mathbb{R}^{n \times d}$ and $x \in \mathbb{R}^d$. If you are given $b = Ax$, under what condition can you find $x$?

\[ n > d \]

\[ \det(A) \neq 0 \quad \text{inverse } A \text{ exists} \]

\[ \text{columns are linearly independent} \]

\[ \text{id.} \]
Consider matrix $A \in \mathbb{R}^{n \times d}$ and $x \in \mathbb{R}^d$. If you are given $b = Ax$, under what condition can you find $x$? When $A$ has full column rank – i.e., all columns are linearly independent.

\[
\begin{bmatrix}
0 \\
0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} = \begin{bmatrix}
b_1 \\
b_2
\end{bmatrix}
\]
Consider matrix \( A \in \mathbb{R}^{n \times d} \) and \( x \in \mathbb{R}^d \). If you are given \( b = Ax \), under what condition can you find \( x \)? When \( A \) has full column rank – i.e., all columns are linearly independent.

\[ n > d \]
Consider matrix $A \in \mathbb{R}^{n \times d}$ and $x \in \mathbb{R}^d$. If you are given $b = Ax$, under what condition can you find $x$? When $A$ has full column rank – i.e., all columns are linearly independent.
Consider matrix $A \in \mathbb{R}^{n \times d}$ and $x \in \mathbb{R}^d$. If you are given $b = Ax$, under what condition can you find $x$? When $A$ has full column rank – i.e., all columns are linearly independent.

**Compressed sensing**: Under what assumptions we can still find $x$ when the number of ‘measurements’ $n$ is smaller than the number of features $d$ (i.e., when $b$ is a compression of $x$)?
The most common assumption of compressed sensing is that $\mathbf{x}$ is $k$-sparse, i.e., has at most $k \ll d$ nonzero entries.
The most common assumption of compressed sensing is that \( x \) is \( k \)-sparse, i.e., has at most \( k \ll d \) nonzero entries.
The most common assumption of compressed sensing is that $x$ is \textit{k-sparse}, i.e., has at most $k \ll d$ nonzero entries.

These types of linear systems have lots of applications.
The most common assumption of compressed sensing is that $x$ is $k$-sparse, i.e., has at most $k \ll d$ nonzero entries.

These types of linear systems have lots of applications.

First: Under what condition can you find $x$ assuming knowledge that it is at most $k$-sparse?
The most common assumption of compressed sensing is that $x$ is $k$-sparse, i.e., has at most $k \ll d$ nonzero entries.

These types of linear systems have lots of applications.

First: Under what condition can you find $x$ assuming knowledge that it is at most $k$-sparse? When every set of $2k$ columns in $A$ is linearly independent – i.e., $A$ has Kruskal rank $2k$. 
**KRUSKAL RANK RECOVERABILITY**

Sufficiency of Kruskal Rank $2k$:

- We want to recover $x$ from $b = Ax$, assuming $x$ is $k$-sparse.
KRUSKAL RANK RECOVERABILITY

Sufficiency of Kruskal Rank $2k$:

- We want to recover $x$ from $b = Ax$, assuming $x$ is $k$-sparse.
- Say there was a different $k$-sparse $x'$ with $Ax' = b$, making recovery impossible.
Sufficiency of Kruskal Rank $2k$:

- We want to recover $x$ from $b = Ax$, assuming $x$ is $k$-sparse.
- Say there was a different $k$-sparse $x'$ with $Ax' = b$, making recovery impossible.
- Then $Ax - Ax' = A(x - x') = 0$.
Sufficiency of Kruskal Rank $2k$:

- We want to recover $\mathbf{x}$ from $\mathbf{b} = \mathbf{A}\mathbf{x}$, assuming $\mathbf{x}$ is $k$-sparse.
- Say there was a different $k$-sparse $\mathbf{x}'$ with $\mathbf{A}\mathbf{x}' = \mathbf{b}$, making recovery impossible.
- Then $\mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{x}' = \mathbf{A}(\mathbf{x} - \mathbf{x}') = 0$. 

![Diagram showing the relationship between A, x, and b](image)
Sufficiency of Kruskal Rank $2k$:

- We want to recover $x$ from $b = Ax$, assuming $x$ is $k$-sparse.
- Say there was a different $k$-sparse $x'$ with $Ax' = b$, making recovery impossible.
- Then $Ax - Ax' = A(x - x') = 0.$
Sufficiency of Kruskal Rank $2k$:

- We want to recover $x$ from $b = Ax$, assuming $x$ is $k$-sparse.
- Say there was a different $k$-sparse $x'$ with $Ax' = b$, making recovery impossible.
- Then $Ax - Ax' = A(x - x') = 0$. 

![Diagram showing $A$, $x$, $x'$, and $0$ equal to each other]
Sufficiency of Kruskal Rank \( 2k \):

- We want to recover \( x \) from \( b = Ax \), assuming \( x \) is \( k \)-sparse.
- Say there was a different \( k \)-sparse \( x' \) with \( Ax' = b \), making recovery impossible.
- Then \( Ax - Ax' = A(x - x') = 0 \).
Sufficiency of Kruskal Rank $2k$:

- We want to recover $x$ from $b = Ax$, assuming $x$ is $k$-sparse.
- Say there was a different $k$-sparse $x'$ with $Ax' = b$, making recovery impossible.
- Then $Ax - Ax' = A(x - x') = 0.$
Sufficiency of Kruskal Rank $2k$:

- We want to recover $x$ from $b = Ax$, assuming $x$ is $k$-sparse.
- Say there was a different $k$-sparse $x'$ with $Ax' = b$, making recovery impossible.
- Then $Ax - Ax' = A(x - x') = 0$. Violates Kruskal rank assumption.

![Diagram showing a set of $\leq 2k$ linearly independent columns]
Sufficiency of Kruskal Rank $2k$:

- We want to recover $x$ from $b = Ax$, assuming $x$ is $k$-sparse.
- Say there was a different $k$-sparse $x'$ with $Ax' = b$, making recovery impossible.
- Then $Ax - Ax' = A(x - x') = 0$. Violates Kruskal rank assumption.

Thus $x$ is the unique $k$-sparse solution to $Ax = b$. 

\[
\begin{align*}
\text{a set of } \leq 2k \text{ linearly independent columns} \\
\end{align*}
\]
To satisfy the Kruskal rank $\geq 2k$ assumption $A$ just needs $2k$ rows (compared with $d$ rows to have full column rank). Can recover a $d$-dimensional $k$-sparse vector $x$ from just $2k$ measurements $b = Ax$. 

$$2k \leq A \begin{bmatrix} \vdots \\ x \\ \vdots \end{bmatrix} = b$$
To satisfy the Kruskal rank $\geq 2k$ assumption $A$ just needs $2k$ rows (compared with $d$ rows to have full column rank). Can recover a $d$-dimensional $k$-sparse vector $x$ from just $2k$ measurements $b = Ax$.

Assuming that $A$ has Kruskal rank $\geq 2k$, how do we actually find the unique $k$-sparse solution $x$ to $Ax = b$?
RECOVERY PROCEDURE

To satisfy the Kruskal rank $\geq 2k$ assumption $A$ just needs $2k$ rows (compared with $d$ rows to have full column rank). Can recover a $d$-dimensional $k$-sparse vector $x$ from just $2k$ measurements $b = Ax$.

Assuming that $A$ has Kruskal rank $\geq 2k$, how do we actually find the unique $k$-sparse solution $\hat{x}$ to $Ax = b$?

$$x = \arg\min_{z \in \mathbb{R}^d: Az = b} \|z\|_0,$$

where $\|z\|_0$ is the number of non-zero entries in $z$. 

$$A \hat{x} = b$$

$$\hat{x} \leq k\text{-sparse}$$
To satisfy the Kruskal rank $\geq 2k$ assumption $A$ just needs $2k$ rows (compared with $d$ rows to have full column rank). Can recover a $d$-dimensional $k$-sparse vector $x$ from just $2k$ measurements $b = Ax$.

Assuming that $A$ has Kruskal rank $\geq 2k$, how do we actually find the unique $k$-sparse solution $x$ to $Ax = b$?

$$x = \arg\min_{z \in \mathbb{R}^d : Az = b} ||z||_0,$$

where $||z||_0$ is the number of non-zero entries in $z$.

This problem seems very difficult to solve. Why?
RECOVERY PROCEDURE

To satisfy the Kruskal rank $\geq 2k$ assumption $A$ just needs $2k$ rows (compared with $d$ rows to have full column rank). Can recover a $d$-dimensional $k$-sparse vector $x$ from just $2k$ measurements $b = Ax$.

Assuming that $A$ has Kruskal rank $\geq 2k$, how do we actually find the unique $k$-sparse solution $x$ to $Ax = b$?

$$x = \arg\min_{z \in \mathbb{R}^d : Az = b} \|z\|_0,$$

where $\|z\|_0$ is the number of non-zero entries in $z$.

This problem seems very difficult to solve. Why? Non-convex.
To satisfy the Kruskal rank $\geq 2k$ assumption $A$ just needs $2k$ rows (compared with $d$ rows to have full column rank). Can recover a $d$-dimensional $k$-sparse vector $x$ from just $2k$ measurements $b = Ax$.

Assuming that $A$ has Kruskal rank $\geq 2k$, how do we actually find the unique $k$-sparse solution $x$ to $Ax = b$?

$$x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0,$$

where $\|z\|_0$ is the number of non-zero entries in $z$.

This problem seems very difficult to solve. Why? Non-convex.

Exponential Time Algorithm:
To satisfy the Kruskal rank $\geq 2k$ assumption $A$ just needs $2k$ rows (compared with $d$ rows to have full column rank). Can recover a $d$-dimensional $k$-sparse vector $x$ from just $2k$ measurements $b = Ax$.

Assuming that $A$ has Kruskal rank $\geq 2k$, how do we actually find the unique $k$-sparse solution $x$ to $Ax = b$?

$$x = \arg\min_{\substack{z \in \mathbb{R}^d: Az = b}} \|z\|_0,$$

where $\|z\|_0$ is the number of non-zero entries in $z$.

This problem seems very difficult to solve. Why? Non-convex.

**Exponential Time Algorithm:** Loop through all $\binom{d}{k} = O(d^k)$ sparsity patterns and find the best $z$ with the given sparsity pattern by solving a normal linear regression problem.
To satisfy the Kruskal rank $\geq 2k$ assumption $A$ just needs $2k$ rows (compared with $d$ rows to have full column rank). Can recover a $d$-dimensional $k$-sparse vector $x$ from just $2k$ measurements $b = Ax$.

Assuming that $A$ has Kruskal rank $\geq 2k$, how do we actually find the unique $k$-sparse solution $x$ to $Ax = b$?

$$x = \arg\min_{z \in \mathbb{R}^d : Az = b} ||z||_0,$$

where $||z||_0$ is the number of non-zero entries in $z$.

This problem seems very difficult to solve. Why? Non-convex.

**Exponential Time Algorithm:** Loop through all $(d \choose k) = O(d^k)$ sparsity patterns and find the best $z$ with the given sparsity pattern by solving a normal linear regression problem.

A major accomplishment of compressed sensing/sparse recovery is to make the above procedure efficient and noise robust.
Example Applications
Example Applications

**Caviat:** Today we will only talk about sparse recovery *without noise* when $\mathbf{Ax} = \mathbf{b}$. In applications, it is important to be able to recover $\mathbf{x}$ from $\mathbf{b}$ with $\mathbf{Ax} = \mathbf{b} + \mathbf{n}$ for some small noise $\mathbf{n}$.
Example Applications

**Caviat:** Today we will only talk about sparse recovery without noise when $Ax = b$. In applications, it is important to be able to recover $x$ from $b$ with $Ax = b + n$ for some small noise $n$.

- The techniques discussed carry over to the noisy setting.
- Generally won’t find $x$ exactly, but up to some good approximation.
In high-dimensional data analysis, you often have a huge number of variables (genetic markers, characteristics of a user, etc.), possibly more than the number of data points.
In high-dimensional data analysis, you often have a huge number of variables (genetic markers, characteristics of a user, etc.), possibly more than the number of data points.

- Believe that just a few important features explain some phenomena (e.g., if a patient is likely to have a certain disease).
In high-dimensional data analysis, you often have a huge number of variables (genetic markers, characteristics of a user, etc.), possibly more than the number of data points.

- Believe that just a few important features explain some phenomena (e.g., if a patient is likely to have a certain disease).
- Want to find a linear regression model \( Ax \approx b \) that only uses a small number of features (\( x \) is sparse).
In high-dimensional data analysis, you often have a huge number of variables (genetic markers, characteristics of a user, etc.), possibly more than the number of data points.

• Believe that just a few important features explain some phenomena (e.g., if a patient is likely to have a certain disease).
• Want to find a linear regression model $\mathbf{A}\mathbf{x} \approx \mathbf{b}$ that only uses a small number of features ($\mathbf{x}$ is sparse).
In high-dimensional data analysis, you often have a huge number of variables (genetic markers, characteristics of a user, etc.), possibly more than the number of data points.

- Believe that just a few important features explain some phenomena (e.g., if a patient is likely to have a certain disease).
- Want to find a linear regression model $\mathbf{Ax} \approx \mathbf{b}$ that only uses a small number of features ($\mathbf{x}$ is sparse).
- Interesting even in the over-constrained case. Often talked about as a different problem than compressed sensing, but very related.
FREQUENT ITEMS COUNTING

Recall: The frequent elements problem asks us to return the $k$ most frequent elements seen in a stream of items.
Recall: The frequent elements problem asks us to return the $k$ most frequent elements seen in a stream of items.

- We saw how to (approximately) solve in $O(k)$ space using Misra-Gries or Count-Min Sketch.
Recall: The frequent elements problem asks us to return the $k$ most frequent elements seen in a stream of items.

- We saw how to (approximately) solve in $O(k)$ space using Misra-Gries or Count-Min Sketch.
- Only work when frequencies are constantly incremented (we see more items over time). But what about when frequencies can be decremented?
Recall: The frequent elements problem asks us to return the $k$ most frequent elements seen in a stream of items.

- We saw how to (approximately) solve in $O(k)$ space using Misra-Gries or Count-Min Sketch.
- Only work when frequencies are constantly incremented (we see more items over time). But what about when frequencies can be decremented?
- E.g., Amazon is monitoring what products people add to their “wishlist” and wants a list of most tagged products. Wishlists can be change over time, and items can be removed, decreasing their frequencies.
Recall: The frequent elements problem asks us to return the $k$ most frequent elements seen in a stream of items.

- We saw how to (approximately) solve in $O(k)$ space using Misra-Gries or Count-Min Sketch.
- Only work when frequencies are constantly incremented (we see more items over time). But what about when frequencies can be decremented?
- E.g., Amazon is monitoring what products people add to their “wishlist” and wants a list of most tagged products. Wishlists can be change over time, and items can be removed, decreasing their frequencies.
- In this setting, the problem is solved with sparse recovery techniques.
FREQUENT ITEMS COUNTING

frequency vector $x$

<table>
<thead>
<tr>
<th>item 1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>item 2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>item n</td>
<td>0</td>
</tr>
</tbody>
</table>
## Frequent Items Counting

A frequency vector \( x \) can be represented as a matrix where:

<table>
<thead>
<tr>
<th></th>
<th>item 1</th>
<th>item 2</th>
<th>...</th>
<th>item n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>

This vector is used to represent the frequency of items in a transactional dataset.
FREQUENT ITEMS COUNTING

frequency vector $\mathbf{x}$

<table>
<thead>
<tr>
<th></th>
<th>item 1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>item 2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$\vdots$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>item n</td>
<td>0</td>
</tr>
</tbody>
</table>
## Frequent Items Counting

A frequency vector $\mathbf{x}$ is a representation of the frequency of items in a dataset. The example shows a frequency vector for a small number of items:

<table>
<thead>
<tr>
<th>item</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>item 1</td>
<td>2</td>
</tr>
<tr>
<td>item 2</td>
<td>0</td>
</tr>
<tr>
<td>item n</td>
<td>0</td>
</tr>
</tbody>
</table>

The diagram visualizes this data with a bar chart, where each item is represented by a bar of its frequency.
FREQUENT ITEMS COUNTING

frequency vector $\mathbf{x}$

<table>
<thead>
<tr>
<th>item 1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>item 2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>item n</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
### Frequency Vector $x$

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item 1</td>
<td>2000</td>
</tr>
<tr>
<td>Item 2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Item n</td>
<td>2</td>
</tr>
</tbody>
</table>
FREQUENT ITEMS COUNTING

<table>
<thead>
<tr>
<th>Item</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item 1</td>
<td>2000</td>
</tr>
<tr>
<td>Item 2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Item n</td>
<td>2</td>
</tr>
</tbody>
</table>

- Storing \( x \) requires \( O(n) \) space. Will instead store \( Ax \) where \( A \in \mathbb{R}^{O(k) \times n} \) is a random matrix.
FREQUENT ITEMS COUNTING

<table>
<thead>
<tr>
<th>Item</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>item 1</td>
<td>2000</td>
</tr>
<tr>
<td>item 2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>item n</td>
<td>2</td>
</tr>
</tbody>
</table>

- Storing $x$ requires $O(n)$ space. Will instead store $Ax$ where $A \in \mathbb{R}^{O(k) \times n}$ is a random matrix.
- $Ax$ can be efficiently updated in a data stream.
FREQUENT ITEMS COUNTING

- Storing $\mathbf{x}$ requires $O(n)$ space. Will instead store $\mathbf{A}\mathbf{x}$ where $\mathbf{A} \in \mathbb{R}^{O(k) \times n}$ is a random matrix.
- $\mathbf{Ax}$ can be efficiently updated in a data stream.
Frequent Items Counting

- Storing $\mathbf{x}$ requires $O(n)$ space. Will instead store $\mathbf{A}\mathbf{x}$ where $\mathbf{A} \in \mathbb{R}^{O(k) \times n}$ is a random matrix.
- $\mathbf{A}\mathbf{x}$ can be efficiently updated in a data stream.
FREQUENT ITEMS COUNTING

![Random Matrix and Frequency Vector]

- Storing $\mathbf{x}$ requires $O(n)$ space. Will instead store $A\mathbf{x}$ where $A \in \mathbb{R}^{O(k) \times n}$ is a random matrix.

- $A\mathbf{x}$ can be efficiently updated in a data stream.
• Storing $\mathbf{x}$ requires $O(n)$ space. Will instead store $\mathbf{Ax}$ where $\mathbf{A} \in \mathbb{R}^{O(k) \times n}$ is a random matrix.

• $\mathbf{Ax}$ can be efficiently updated in a data stream.
Frequent Items Counting

- Storing \( \mathbf{x} \) requires \( O(n) \) space. Will instead store \( \mathbf{A}\mathbf{x} \) where \( \mathbf{A} \in \mathbb{R}^{O(k) \times n} \) is a random matrix.
- \( \mathbf{A}\mathbf{x} \) can be efficiently updated in a data stream.
FREQUENT ITEMS COUNTING

- Storing $\mathbf{x}$ requires $O(n)$ space. Will instead store $\mathbf{Ax}$ where $\mathbf{A} \in \mathbb{R}^{O(k) \times n}$ is a random matrix.
- $\mathbf{Ax}$ can be efficiently updated in a data stream.
FREQUENT ITEMS COUNTING

- Storing $\mathbf{x}$ requires $O(n)$ space. Will instead store $\mathbf{Ax}$ where $\mathbf{A} \in \mathbb{R}^{O(k) \times n}$ is a random matrix.
- $\mathbf{Ax}$ can be efficiently updated in a data stream.
- If there are a $k$ heavy items, $\mathbf{x}$ is approximately $k$-sparse.
- Estimating the large entries of $\mathbf{x}$ (the counts of the most frequent items) from the compression $\mathbf{Ax}$ is exactly sparse recovery.
Many of the most important applications of sparse recovery are in imaging and signal processing.

- Many signals are sparse in some basis (Fourier, wavelet, etc.).
- Using sparse recovery techniques, an $n$ pixel image/$n$ point signal can thus be recovered from many fewer than $n$ measurements.
- Efficient MRI imaging, remote sensing for oil exploration, GPS synchronization, power efficient cameras, etc.
Many of the most important applications of sparse recovery are in imaging and signal processing.

- Many signals are sparse in some basis (Fourier, wavelet, etc.).
- Using sparse recovery techniques, an $n$ pixel image/$n$ point signal can thus be recovered from many fewer than $n$ measurements.
- Efficient MRI imaging, remote sensing for oil exploration, GPS synchronization, power efficient cameras, etc.

In general, there are a lot of practical complexities here. So everything I say is a major oversimplification.
Discrete Fourier Transform: For a discrete signal (aka a vector) $x \in \mathbb{R}^n$, its discrete Fourier transform is denoted $\hat{x} \in \mathbb{C}^n$ and given by $\hat{x} = Fx$, where $F \in \mathbb{C}^{n \times n}$ is the discrete Fourier transform matrix.
Discrete Fourier Transform: For a discrete signal (aka a vector) \( x \in \mathbb{R}^n \), its discrete Fourier transform is denoted \( \hat{x} \in \mathbb{C}^n \) and given by \( \hat{x} = Fx \), where \( F \in \mathbb{C}^{n \times n} \) is the discrete Fourier transform matrix.

For many natural signals \( \hat{x} \) is approximately sparse: a few dominant frequencies in a recording, superposition of a few radio transmitters sending at different frequencies, etc.
When the Fourier transform $\hat{x}$ is sparse, can recover $x$ from few measurements using sparse recovery.
When the Fourier transform $\hat{x}$ is sparse, can recover $x$ from few measurements using sparse recovery.
When the Fourier transform $\hat{x}$ is sparse, can recover $x$ from few measurements using sparse recovery.
When the Fourier transform $\hat{x}$ is sparse, can recover $x$ from few measurements using sparse recovery.
When the Fourier transform $\hat{x}$ is sparse, can recover $x$ from few measurements using sparse recovery.

Translates to big savings in acquisition costs, the number of sensors required, etc.
Back to Algorithms
We would like to recover $k$-sparse $x$ from measurements $b = Ax$ by solving the non-convex optimization problem:

$$x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$$

Works if $A$ has Kruskal rank $\geq 2k$, but very hard computationally.
We would like to recover $k$-sparse $x$ from measurements $b = Ax$ by solving the non-convex optimization problem:

$$x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$$

Works if $A$ has Kruskal rank $\geq 2k$, but very hard computationally. **Convex Relaxation:** A very common technique. Just ‘relax’ the problem to be convex.

$$x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_1 \quad \text{where} \quad \|z\|_1 = \sum_{i=1}^{d} |z(i)|.$$
We would like to recover \( k \)-sparse \( x \) from measurements \( b = Ax \) by solving the non-convex optimization problem:

\[
 x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0
\]

Works if \( A \) has Kruskal rank \( \geq 2k \), but very hard computationally.

**Convex Relaxation:** A very common technique. Just ‘relax’ the problem to be convex. **Basis Pursuit:**

\[
 x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_1 \quad \text{where} \quad \|z\|_1 = \sum_{i=1}^{d} |z(i)|.
\]
CONVEX RELAXATION

We would like to recover \( k \)-sparse \( x \) from measurements \( b = Ax \) by solving the non-convex optimization problem:

\[
 x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0
\]

Works if \( A \) has Kruskal rank \( \geq 2k \), but very hard computationally. **Convex Relaxation:** A very common technique. Just ‘relax’ the problem to be convex. **Basis Pursuit:**

\[
 x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_1 \quad \text{where} \quad \|z\|_1 = \sum_{i=1}^{d} |z(i)|.
\]

What is one algorithm we have learned for solving this problem?
We would like to recover $k$-sparse $x$ from measurements $b = Ax$ by solving the non-convex optimization problem:

$$x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$$

Works if $A$ has Kruskal rank $\geq 2k$, but very hard computationally. **Convex Relaxation:** A very common technique. Just ‘relax’ the problem to be convex. **Basis Pursuit:**

$$x = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_1 \quad \text{where} \quad \|z\|_1 = \sum_{i=1}^{d} |z(i)|.$$

What is one algorithm we have learned for solving this problem?

- Projected gradient descent – convex objective function and convex constraint set.
We would like to recover $k$-sparse $\mathbf{x}$ from measurements $\mathbf{b} = \mathbf{A}\mathbf{x}$ by solving the non-convex optimization problem:

$$\mathbf{x} = \arg \min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_0$$

Works if $\mathbf{A}$ has Kruskal rank $\geq 2k$, but very hard computationally.

**Convex Relaxation:** A very common technique. Just ‘relax’ the problem to be convex. **Basis Pursuit:**

$$\mathbf{x} = \arg \min_{\mathbf{z} \in \mathbb{R}^d : \mathbf{A}\mathbf{z} = \mathbf{b}} \|\mathbf{z}\|_1 \quad \text{where} \quad \|\mathbf{z}\|_1 = \sum_{i=1}^{d} |\mathbf{z}(i)|.$$
Why should we hope that the basis pursuit solution returns the unique $k$-sparse $x$ with $Ax = b$? The minimizer $z^*$ will have small $\ell_1$ norm but why would it even be sparse?

$$\arg\min_{z \in \mathbb{R}^d : Az = b} \|z\|_1 \quad \text{vs.} \quad \arg\min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$$
Why should we hope that the basis pursuit solution returns the unique $k$-sparse $x$ with $Ax = b$? The minimizer $z^*$ will have small $\ell_1$ norm but why would it even be sparse?

$$\arg\min_{z \in \mathbb{R}^d : Az = b} \|z\|_1 \quad \text{vs.} \quad \arg\min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$$

Assume that $n = 1$, $d = 2$, $k = 1$. So $A \in \mathbb{R}^{1 \times 2}$ and $x \in \mathbb{R}^2$ is 1-sparse.
BASIS PURSUIT

Why should we hope that the basis pursuit solution returns the unique $k$-sparse $x$ with $Ax = b$? The minimizer $z^*$ will have small $\ell_1$ norm but why would it even be sparse?

$$\arg\min_{z \in \mathbb{R}^d : Az = b} \|z\|_1 \quad \text{vs.} \quad \arg\min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$$

Assume that $n = 1$, $d = 2$, $k = 1$. So $A \in \mathbb{R}^{1 \times 2}$ and $x \in \mathbb{R}^2$ is 1-sparse.
Why should we hope that the basis pursuit solution returns the unique $k$-sparse $x$ with $Ax = b$? The minimizer $z^*$ will have small $\ell_1$ norm but why would it even be sparse?

$$\arg \min_{z \in \mathbb{R}^d : Az = b} ||z||_1 \quad \text{vs.} \quad \arg \min_{z \in \mathbb{R}^d : Az = b} ||z||_0$$

Assume that $n = 1, d = 2, k = 1$. So $A \in \mathbb{R}^{1 \times 2}$ and $x \in \mathbb{R}^2$ is 1-sparse.
Why should we hope that the basis pursuit solution returns the unique $k$-sparse $x$ with $Ax = b$? The minimizer $z^*$ will have small $\ell_1$ norm but why would it even be sparse?

$$\arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_1 \quad \text{vs.} \quad \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$$

Assume that $n = 1$, $d = 2$, $k = 1$. So $A \in \mathbb{R}^{1 \times 2}$ and $x \in \mathbb{R}^2$ is 1-sparse.
BASIS PURSUIT

Why should we hope that the basis pursuit solution returns the unique $k$-sparse $x$ with $Ax = b$? The minimizer $z^*$ will have small $\ell_1$ norm but why would it even be sparse?

$$\arg\min_{z \in \mathbb{R}^d: Az = b} \|z\|_1 \quad \text{vs.} \quad \arg\min_{z \in \mathbb{R}^d: Az = b} \|z\|_0$$

Assume that $n = 1$, $d = 2$, $k = 1$. So $A \in \mathbb{R}^{1 \times 2}$ and $x \in \mathbb{R}^2$ is 1-sparse.

- Optimization solution will be on a corner (i.e., sparse), unless $Az = b$ has slope 1.
BASIS PURSUIT

Why should we hope that the basis pursuit solution returns the unique $k$-sparse $x$ with $Ax = b$? The minimizer $z^*$ will have small $\ell_1$ norm but why would it even be sparse?

$$\arg\min_{z\in\mathbb{R}^d:Az=b} ||z||_1 \quad \text{vs.} \quad \arg\min_{z\in\mathbb{R}^d:Az=b} ||z||_0$$

Assume that $n = 1$, $d = 2$, $k = 1$. So $A \in \mathbb{R}^{1\times 2}$ and $x \in \mathbb{R}^2$ is 1-sparse.

- Optimization solution will be on a corner (i.e., sparse), unless $Az = b$ has slope 1.
- Similar intuition to the LASSO method.
BASIS PURSUIT

Why should we hope that the basis pursuit solution returns the unique \(k\)-sparse \(x\) with \(Ax = b\)? The minimizer \(z^*\) will have small \(\ell_1\) norm but why would it even be sparse?

\[
\arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_1 \quad \text{vs.} \quad \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0
\]

Assume that \(n = 1, d = 2, k = 1\). So \(A \in \mathbb{R}^{1 \times 2}\) and \(x \in \mathbb{R}^2\) is 1-sparse.

- Optimization solution will be on a corner (i.e., sparse), unless \(Az = b\) has slope 1.
- Similar intuition to the LASSO method.
- Does not hold if e.g., the \(\ell_2\) norm is used.
BASIS PURSUIT

Why should we hope that the basis pursuit solution returns the unique \( k \)-sparse \( \mathbf{x} \) with \( \mathbf{A} \mathbf{x} = \mathbf{b} \)? The minimizer \( \mathbf{z}^* \) will have small \( \ell_1 \) norm but why would it even be sparse?

\[
\arg \min_{\mathbf{z} \in \mathbb{R}^d: \mathbf{A} \mathbf{z} = \mathbf{b}} \| \mathbf{z} \|_1 \quad \text{vs.} \quad \arg \min_{\mathbf{z} \in \mathbb{R}^d: \mathbf{A} \mathbf{z} = \mathbf{b}} \| \mathbf{z} \|_0
\]

Assume that \( n = 1, d = 2, k = 1 \). So \( \mathbf{A} \in \mathbb{R}^{1 \times 2} \) and \( \mathbf{x} \in \mathbb{R}^2 \) is 1-sparse.

- Optimization solution will be on a corner (i.e., sparse), unless \( \mathbf{A} \mathbf{z} = \mathbf{b} \) has slope 1.
- Similar intuition to the LASSO method.
- Does not hold if e.g., the \( \ell_2 \) norm is used.
Can prove that basis pursuit outputs the exact $k$-sparse solution $x$ with $Ax = b$ (same as $\arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$)

- Requires a strengthening of the Kruskal rank $\geq 2k$ assumption (that still holds in all the applications discussed).
Can prove that basis pursuit outputs the exact $k$-sparse solution $x$ with $Ax = b$ (same as $\arg\min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$)

- Requires a strengthening of the Kruskal rank $\geq 2k$ assumption (that still holds in all the applications discussed).

**Definition:** $A \in \mathbb{R}^{n \times d}$ has the $(m, \epsilon)$ restricted isometry property (is $(m, \epsilon)$-RIP) if for all $m$-sparse vectors $x$:

$$(1 - \epsilon)\|x\|_2 \leq \|Ax\|_2 \leq (1 + \epsilon)\|x\|_2$$
Can prove that basis pursuit outputs the exact $k$-sparse solution $x$ with $Ax = b$ (same as $\arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_0$)

- Requires a strengthening of the Kruskal rank $\geq 2k$ assumption (that still holds in all the applications discussed).

**Definition:** $A \in \mathbb{R}^{n \times d}$ has the $(m, \epsilon)$ restricted isometry property (is $(m, \epsilon)$-RIP) if for all $m$-sparse vectors $x$:

$$(1 - \epsilon)\|x\|_2 \leq \|Ax\|_2 \leq (1 + \epsilon)\|x\|_2$$

**Theorem:** If $A$ is $(3k, \epsilon)$-RIP for small enough constant $\epsilon$, then $z^* = \arg \min_{z \in \mathbb{R}^d : Az = b} \|z\|_1$ is equal to the unique $k$-sparse $x$ with $Ax = b$ (i.e., basis pursuit solves the sparse recovery problem).
Questions?