ALGORITHMS FOR DICTIONARY LEARNING

ANKUR MOITRA

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

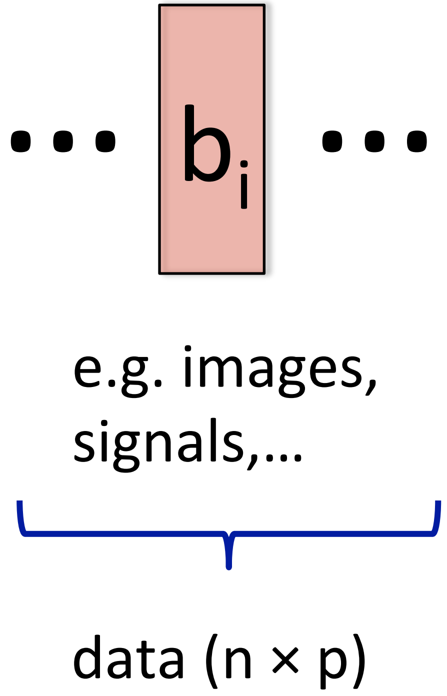
SPARSE REPRESENTATIONS

SPARSE REPRESENTATIONS

Many data-types are sparse in an appropriately chosen basis:

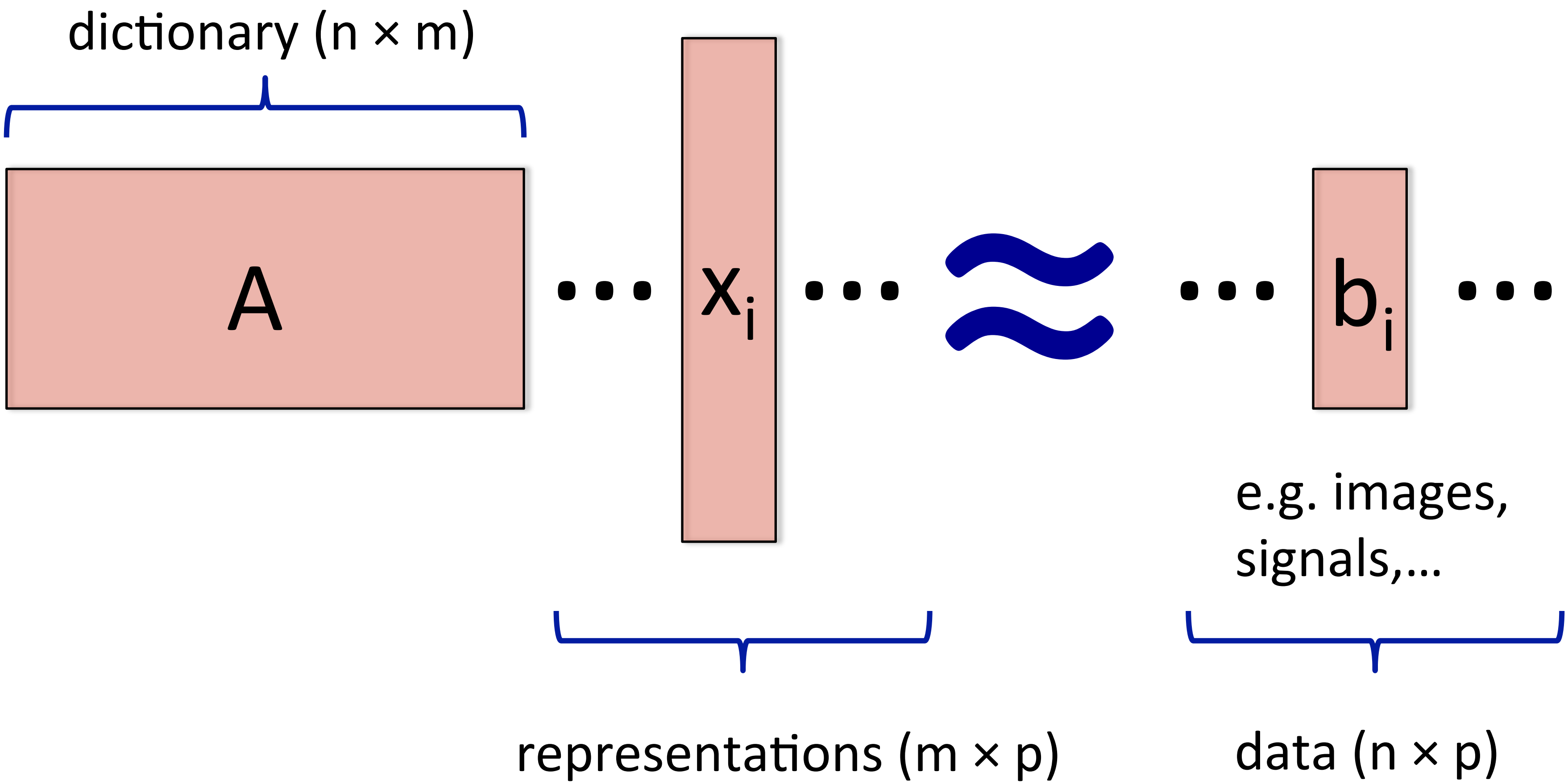
SPARSE REPRESENTATIONS

Many data-types are sparse in an appropriately chosen basis:



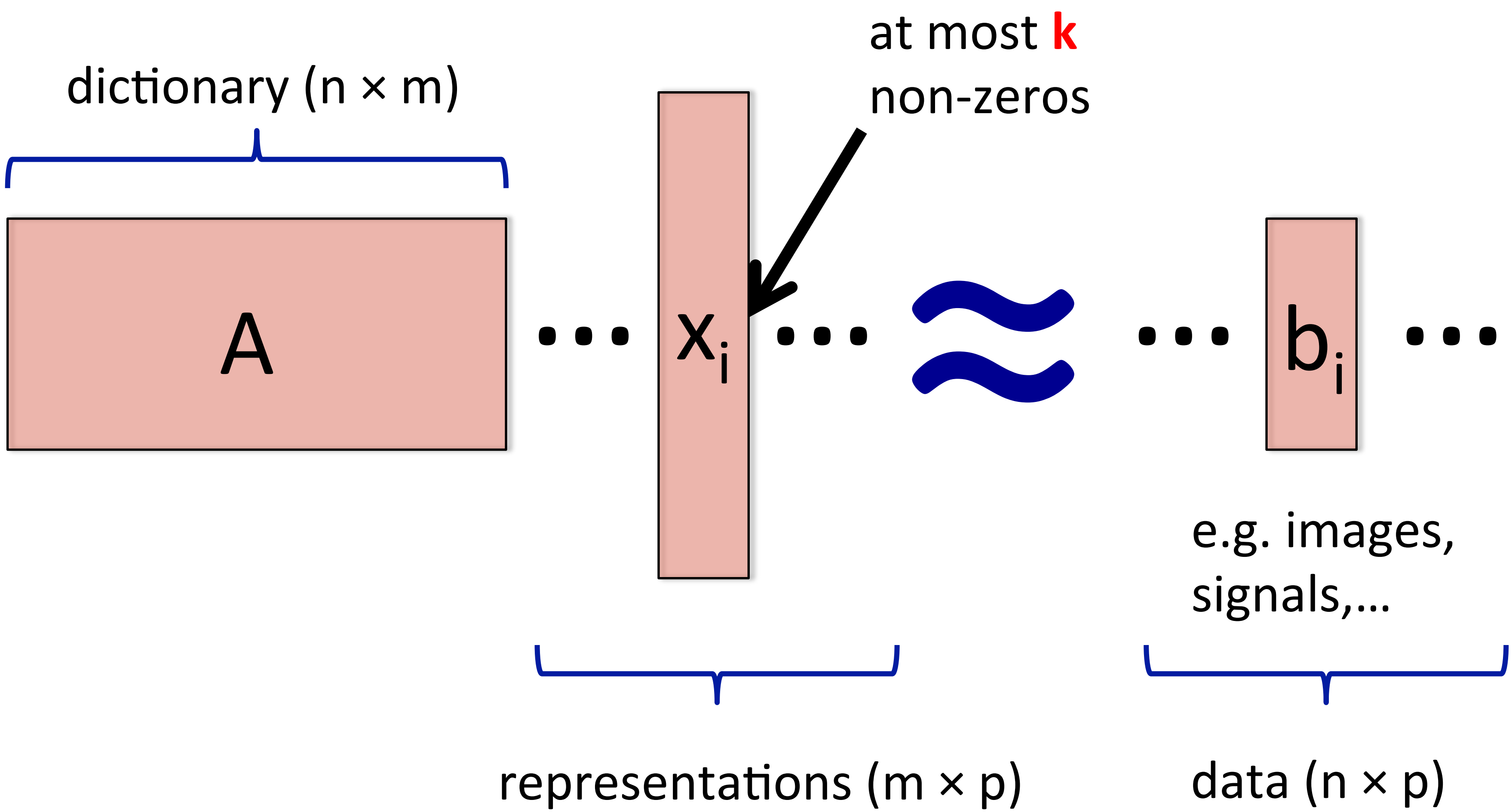
SPARSE REPRESENTATIONS

Many data-types are sparse in an appropriately chosen basis:



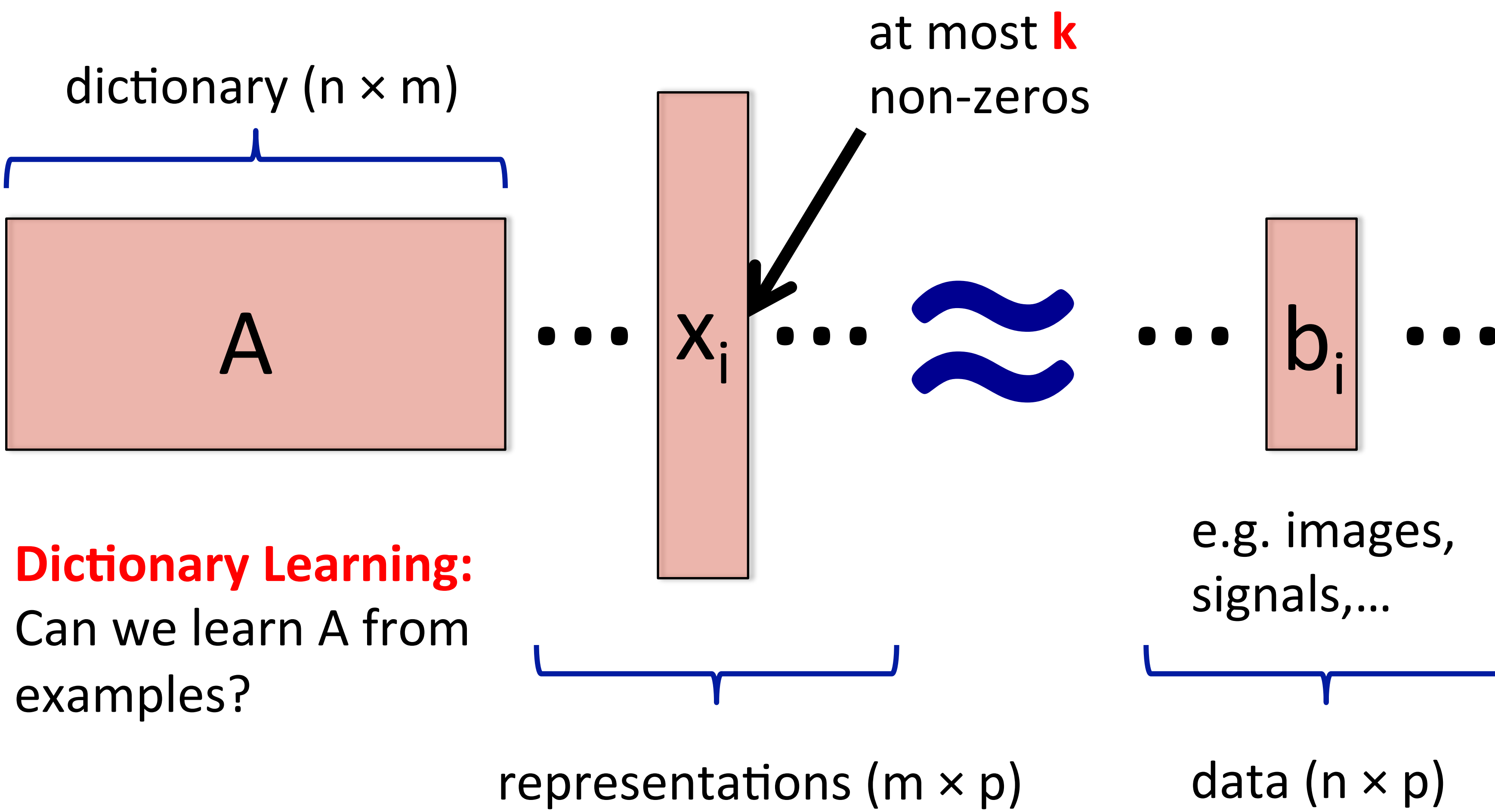
SPARSE REPRESENTATIONS

Many data-types are sparse in an appropriately chosen basis:



SPARSE REPRESENTATIONS

Many data-types are sparse in an appropriately chosen basis:



Dictionary Learning:
Can we learn A from examples?

APPLICATIONS OF DICTIONARY LEARNING

a.k.a. sparse coding

APPLICATIONS OF DICTIONARY LEARNING

a.k.a. sparse coding

Signal Processing/Statistics:

- De-noising, edge-detection, super-resolution
- Block compression for images/video

APPLICATIONS OF DICTIONARY LEARNING

a.k.a. sparse coding

Signal Processing/Statistics:

- De-noising, edge-detection, super-resolution
- Block compression for images/video

Machine Learning:

- Sparsity as a **regularizer** to prevent over-fitting
- Learned sparse reps. play a key role in deep-learning

APPLICATIONS OF DICTIONARY LEARNING

a.k.a. sparse coding

Signal Processing/Statistics:

- De-noising, edge-detection, super-resolution
- Block compression for images/video

Machine Learning:

- Sparsity as a **regularizer** to prevent over-fitting
- Learned sparse reps. play a key role in deep-learning

Computational Neuroscience (Olshausen-Field 1997):

- Applied to natural images yields filters with same qualitative properties as receptive field in V1

OUTLINE

Are there efficient algorithms for dictionary learning?

Introduction

- Origins of Sparse Recovery
- A Stochastic Model; Our Results

Provable Algorithms via Overlapping Clustering

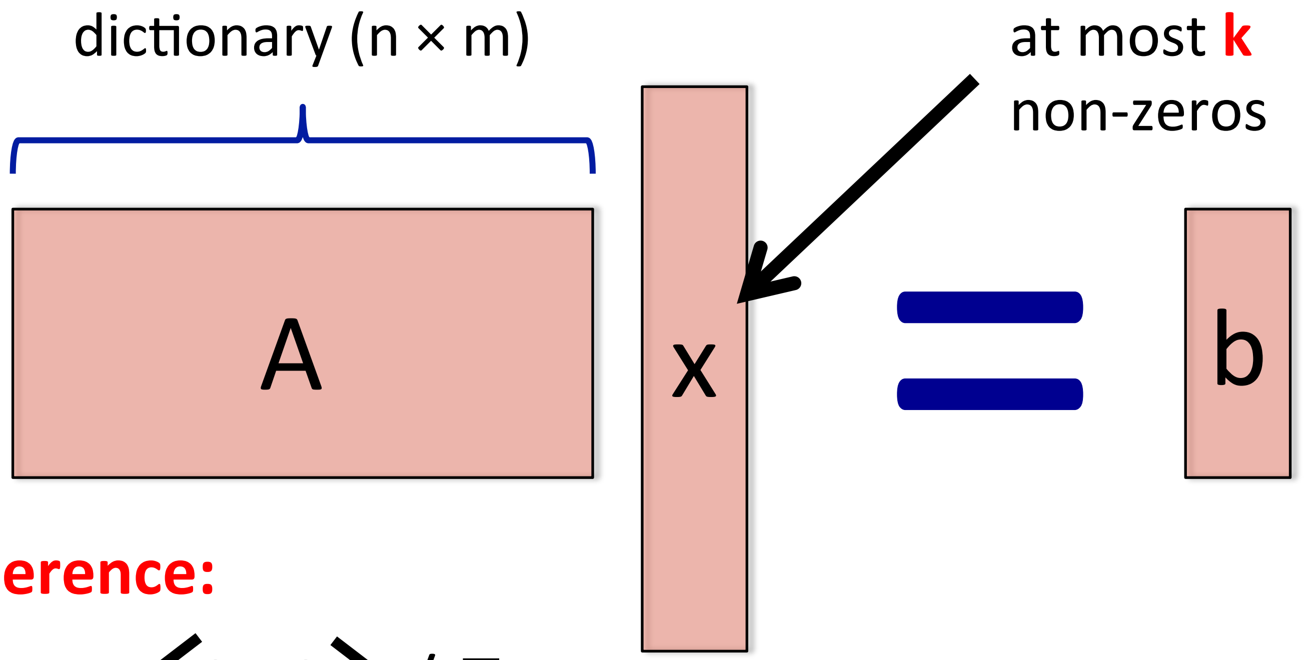
- Uncertainty Principles
- Reformulation as Overlapping Clustering

Analyzing Alternating Minimization

- Gradient Descent on Non-Convex Fctns

ORIGINS OF SPARSE RECOVERY

Donoho-Stark, Donoho-Huo, Gribonval-Nielsen, Donoho-Elad:

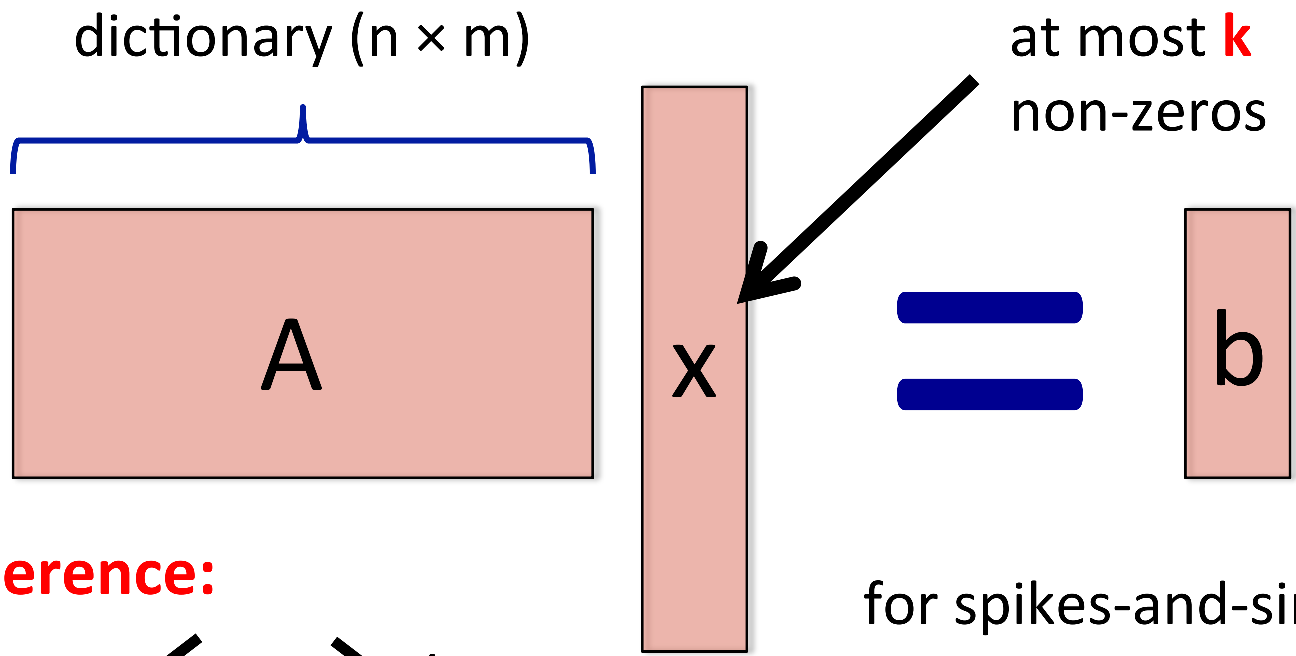


Incoherence:

$\mu = \max_{i \neq j} \langle A_i, A_j \rangle / \sqrt{n}$

ORIGINS OF SPARSE RECOVERY

Donoho-Stark, Donoho-Huo, Gribonval-Nielsen, Donoho-Elad:



Incoherence:

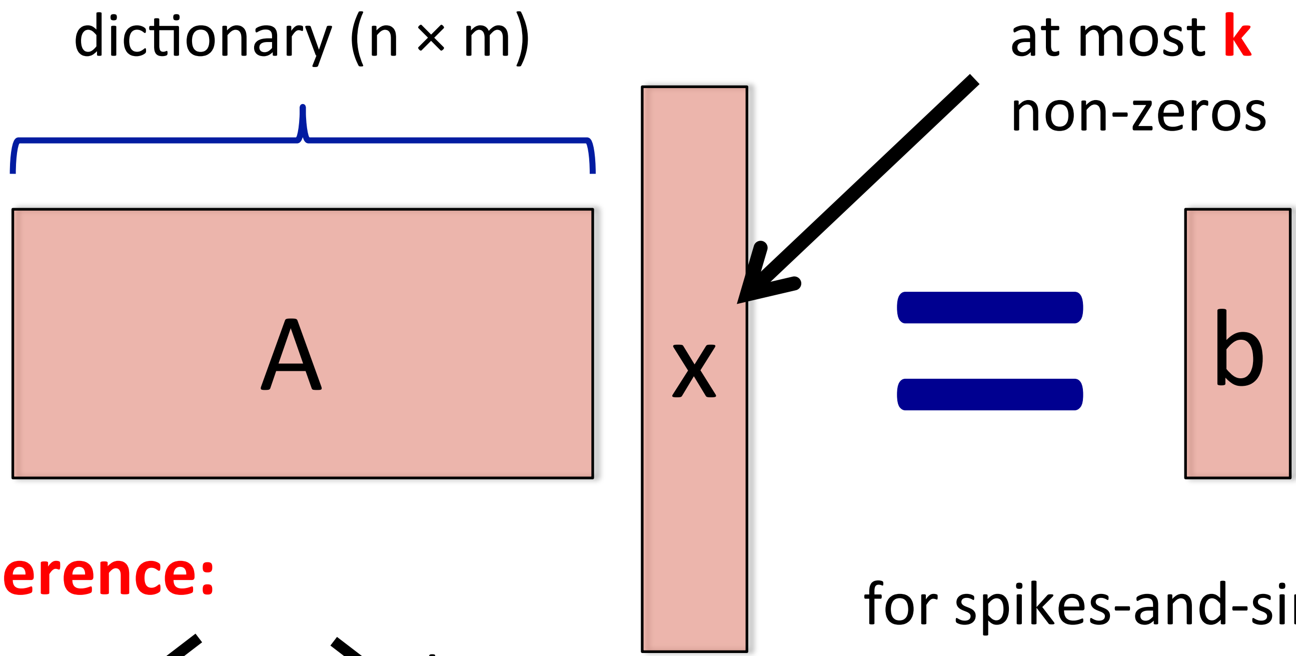
$\mu = \max_{i \neq j} \langle A_i, A_j \rangle / \sqrt{n}$

for spikes-and-sines $\mu = 1$

ORIGINS OF SPARSE RECOVERY

Donoho-Stark, Donoho-Huo, Gribonval-Nielsen, Donoho-Elad:

- If $k \leq \sqrt{n} / 2\mu$ then x is the sparsest solution to the linear system, and can be found with l_1-minimization



Incoherence:

$\mu = \max_{i \neq j} \langle A_i, A_j \rangle / \sqrt{n}$

for spikes-and-sines $\mu = 1$

THE FULL RANK CASE

Are there efficient algorithms for dictionary learning?

Case #1: A has full column rank

THE FULL RANK CASE

Are there efficient algorithms for dictionary learning?

Case #1: A has full column rank

Theorem [Spielman, Wang, Wright '13]: There is a poly. time algorithm to exactly learn A when it has full column rank, for $k \approx \sqrt{n}$ (hence $m \leq n$)

THE FULL RANK CASE

Are there efficient algorithms for dictionary learning?

Case #1: A has full column rank

Theorem [Spielman, Wang, Wright '13]: There is a poly. time algorithm to exactly learn A when it has full column rank, for $k \approx \sqrt{n}$ (hence $m \leq n$)

Approach: find the rows of A^{-1}, using L_1-minimization

THE FULL RANK CASE

Are there efficient algorithms for dictionary learning?

Case #1: A has full column rank

Theorem [Spielman, Wang, Wright '13]: There is a poly. time algorithm to exactly learn A when it has full column rank, for $k \approx \sqrt{n}$ (hence $m \leq n$)

Approach: find the rows of A^{-1}, using L_1-minimization

Stochastic Model:

- unknown dictionary A
- generate x with support of size k u.a.r., choose non-zero values independently, observe $b = Ax$

Notation: $AX = B$, where the columns of B, X are samples and their representations respectively

Notation: AX = B, where the columns of B, X are samples and their representations respectively

Claim: row-span(B) = row-span(X)

Notation: $AX = B$, where the columns of B, X are samples and their representations respectively

Claim: row-span(B) = row-span(X)

Claim: The sparsest vectors in row-span(X) (or B) are the X

Notation: AX = B, where the columns of B, X are samples and their representations respectively

Claim: row-span(B) = row-span(X)

Claim: The sparsest vectors in row-span(X) (or B) are the X

Can we find the sparsest vector in row-span(X)?

Notation: AX = B, where the columns of B, X are samples and their representations respectively

Claim: row-span(B) = row-span(X)

Claim: The sparsest vectors in row-span(X) (or B) are the X

Can we find the sparsest vector in row-span(X)?

Approach #1:

$$\text{(P0):} \quad \min \; \|w^T B\|_0 \quad \text{s.t. } w \neq 0$$

Notation: $AX = B$, where the columns of B, X are samples and their representations respectively

Claim: row-span(B) = row-span(X)

Claim: The sparsest vectors in row-span(X) (or B) are the X

Can we find the sparsest vector in row-span(X)?

Approach #1: NP-hard

$$(P0): \quad \min \; \|w^T B\|_1 \quad \text{s.t. } w \neq 0$$

Notation: $AX = B$, where the columns of B, X are samples and their representations respectively

Claim: row-span(B) = row-span(X)

Claim: The sparsest vectors in row-span(X) (or B) are the X

Can we find the sparsest vector in row-span(X)?

Approach #2: L_1-relaxation

$$(P1): \quad \min \ \|w^T B\|_1 \quad \text{s.t. } w^T r = 1$$

where we will set r later…

$$(P1): \quad \min \; ||w^T B||_1 \quad s.t. \; w^T r = 1$$

$$\text{(P1):} \quad \min \ ||w^T B||_1 \quad \text{s.t. } w^T r = 1$$

Consider the bijection $z = A^T w$, and set $c = A^{-1} r$.

$$(P1): \quad \min \; ||w^T B||_1 \quad \text{s.t. } w^T r = 1$$

Consider the bijection $z = A^T w$, and set $r = Ac$. We get:

$$(P1): \quad \min \; ||w^T AX||_1 \quad \text{s.t. } w^T Ac = 1$$

$$(P1): \quad \min \ ||w^T B||_1 \quad \text{s.t. } w^T r = 1$$

Consider the bijection $z = A^T w$, and set $r = Ac$. We get:

$$(P1): \quad \min \ ||w^T AX||_1 \quad \text{s.t. } w^T Ac = 1$$

This is equivalent to:

$$(Q1): \quad \min \ ||z^T X||_1 \quad \text{s.t. } z^T c = 1$$

$$(P1): \quad \min \; ||w^T B||_1 \quad \text{s.t. } w^T r = 1$$

Consider the bijection $z = A^T w$, and set $r = Ac$. We get:

$$(P1): \quad \min \; ||w^T A X||_1 \quad \text{s.t. } w^T A c = 1$$

This is equivalent to:

$$(Q1): \quad \min \; ||z^T X||_1 \quad \text{s.t. } z^T c = 1$$

Set r = column of B, then $c = A^{-1} r$ = column of X

$$(P1): \quad \min \ ||w^T B||_1 \quad \text{s.t. } w^T r = 1$$

Consider the bijection $z = A^T w$, and set $r = Ac$. We get:

$$(P1): \quad \min \ ||w^T AX||_1 \quad \text{s.t. } w^T Ac = 1$$

This is equivalent to:

$$(Q1): \quad \min \ ||z^T X||_1 \quad \text{s.t. } z^T c = 1$$

Set r = column of B, then $c = A^{-1} r$ = column of X

Claim: If c has a strictly largest coordinate ($|c_i| > |c_j|$ for $j \neq i$) in absolute value, then whp the soln to (Q1) is e_i

$$\text{(P1):} \quad \min \ ||w^T B||_1 \quad \text{s.t. } w^T r = 1$$

Consider the bijection $z = A^T w$, and set $r = Ac$. We get:

$$\text{(P1):} \quad \min \ ||w^T AX||_1 \quad \text{s.t. } w^T Ac = 1$$

Claim: Then the soln to (P1) is the i^{th} row of X

This is equivalent to:

$$\text{(Q1):} \quad \min \ ||z^T X||_1 \quad \text{s.t. } z^T c = 1$$

Set r = column of B, then $c = A^{-1}r$ = column of X

Claim: If c has a strictly largest coordinate ($|c_i| > |c_j|$ for $j \neq i$) in absolute value, then whp the soln to (Q1) is e_i

Notation: $AX = B$, where the columns of B, X are samples and their representations respectively

Claim: row-span(B) = row-span(X)

Claim: The sparsest vectors in row-span(X) (or B) are the X

Can we find the sparsest vector in row-span(X)?

Approach #2: L_1-relaxation

$$\text{(P1):} \quad \min \ ||w^T B||_1 \quad \text{s.t. } w^T r = 1$$

Notation: $AX = B$, where the columns of B, X are samples and their representations respectively

Claim: row-span(B) = row-span(X)

Claim: The sparsest vectors in row-span(X) (or B) are the X

Can we find the sparsest vector in row-span(X)?

Approach #2: L_1-relaxation

$$(P1): \quad \min \ ||w^T B||_1 \quad \text{s.t. } w^T r = 1$$

Hence we can find the rows of X, and solve for A

THE OVERCOMPLETE CASE

What about overcomplete dictionaries?

(more expressive)

Case #2: A is incoherent

THE OVERCOMPLETE CASE

What about overcomplete dictionaries? **(more expressive)**

Case #2: A is incoherent

Theorem [Arora, Ge, Moitra '13]: There is an algorithm to learn A within ε if it is n by m and μ-incoherent for

$$k \approx \min(\sqrt{n}/\mu \log n, m^{1/2-\eta})$$

The running time and sample complexity are poly(n,m,log 1/ε)

THE OVERCOMPLETE CASE

What about overcomplete dictionaries? **(more expressive)**

Case #2: A is incoherent

Theorem [Arora, Ge, Moitra '13]: There is an algorithm to learn A within ε if it is n by m and μ-incoherent for

$$k \approx \min(\sqrt{n}/\mu \log n, m^{1/2-\eta})$$

The running time and sample complexity are poly(n,m,log 1/ε)

Approach: learn the support of the representations X = [… x …] first, by solving an **overlapping clustering** problem…

THE OVERCOMPLETE CASE

What about overcomplete dictionaries? **(more expressive)**

Case #2: A is incoherent

Theorem [Arora, Ge, Moitra '13]: There is an algorithm to learn A within ε if it is n by m and μ-incoherent for

$$k \approx \min(\sqrt{n}/\mu \log n,\ m^{1/2-\eta})$$

The running time and sample complexity are poly(n,m,log 1/ε)

Approach: learn the support of the representations X = [… x …] first, by solving an **overlapping clustering** problem…

Theorem [Agarwal et al '13]: There is a poly. time algorithm to learn A if it is μ-incoherent for $k \approx n^{1/4}/\mu$

THE MODEL

What about overcomplete dictionaries? **(more expressive)**

Case #2: A is incoherent

THE MODEL

What about overcomplete dictionaries?

(more expressive)

Case #2: A is incoherent

Theorem [Barak, Kelner, Steurer '14]: There is a quasi-poly. time algorithm to learn A within any constant A if it is μ-incoherent for $k \approx n^{1-\eta}$ using the sum-of-squares hierarchy

THE MODEL

What about overcomplete dictionaries? **(more expressive)**

Case #2: A is incoherent

Theorem [Barak, Kelner, Steurer '14]: There is a quasi-poly. time algorithm to learn A within any constant A if it is μ-incoherent for $k \approx n^{1-\eta}$ using the sum-of-squares hierarchy

Approach: find y that approximately maximizes $\mathbf{E}[|b^T y|^4]$ via a poly-logarithmic number of rounds; it is close to a coln of A

OUTLINE

Are there efficient algorithms for dictionary learning?

Introduction

- Origins of Sparse Recovery
- A Stochastic Model; Our Results

Provable Algorithms via Overlapping Clustering

- Uncertainty Principles
- Reformulation as Overlapping Clustering

Analyzing Alternating Minimization

- Gradient Descent on Non-Convex Fctns

UNCERTAINTY PRINCIPLES

UNCERTAINTY PRINCIPLES

Claim: Given A, b and k it is **NP**-hard to decide if there is a k-sparse x such that Ax = b

UNCERTAINTY PRINCIPLES

Claim: Given A, b and k it is **NP**-hard to decide if there is a k-sparse x such that Ax = b

Why is this easier for incoherent dictionaries?

UNCERTAINTY PRINCIPLES

Claim: Given A, b and k it is **NP**-hard to decide if there is a k-sparse x such that Ax = b

Why is this easier for incoherent dictionaries?

Uncertainty Principle: If A is μ-incoherent then

$$\langle Ay, Ax \rangle \approx \langle y, x \rangle$$

provided that x and y are k-sparse, for $k \leq \sqrt{n}/2\mu$

UNCERTAINTY PRINCIPLES

Claim: Given A, b and k it is **NP**-hard to decide if there is a k-sparse x such that Ax = b

Why is this easier for incoherent dictionaries?

Uncertainty Principle: If A is μ-incoherent then

$$\langle Ay, Ax \rangle \approx \langle y, x \rangle$$

provided that x and y are k-sparse, for $k \leq \sqrt{n}/2\mu$

Proof: A^TA restricted to the support of x and y is k × k and

$$|(A^TA)_{i,j}| = \begin{cases} 1 & \text{if } i = j \\ \leq \mu/\sqrt{n} & \text{if } i \neq j \end{cases}$$

UNCERTAINTY PRINCIPLES

Claim: Given A, b and k it is **NP**-hard to decide if there is a k-sparse x such that Ax = b

Why is this easier for incoherent dictionaries?

Uncertainty Principle: If A is μ-incoherent then

$$\langle Ay, Ax \rangle \approx \langle y, x \rangle$$

provided that x and y are k-sparse, for $k \leq \sqrt{n}/2\mu$

Proof: A^TA restricted to the support of x and y is k × k and

$$|(A^TA)_{i,j}| = \begin{cases} 1 & \text{if } i = j \\ \leq \mu/\sqrt{n} & \text{if } i \neq j \end{cases}$$

Then use Gershgorin's Disk Thm…

UNCERTAINTY PRINCIPLES

Claim: Given A, b and k it is **NP**-hard to decide if there is a k-sparse x such that Ax = b

Why is this easier for incoherent dictionaries?

Uncertainty Principle: If A is μ-incoherent then

$$\langle Ay, Ax \rangle \approx \langle y, x \rangle$$

provided that x and y are k-sparse, for $k \leq \sqrt{n}/2\mu$

UNCERTAINTY PRINCIPLES

Claim: Given A, b and k it is **NP**-hard to decide if there is a k-sparse x such that Ax = b

Why is this easier for incoherent dictionaries?

Uncertainty Principle: If A is μ-incoherent then

$$\langle Ay, Ax \rangle \approx \langle y, x \rangle$$

provided that x and y are k-sparse, for $k \leq \sqrt{n}/2\mu$

This principle can be used to establish uniqueness for sparse recovery, and things like…

"b cannot be sparse in both standard and Fourier basis"

A PAIR-WISE TEST

A PAIR-WISE TEST

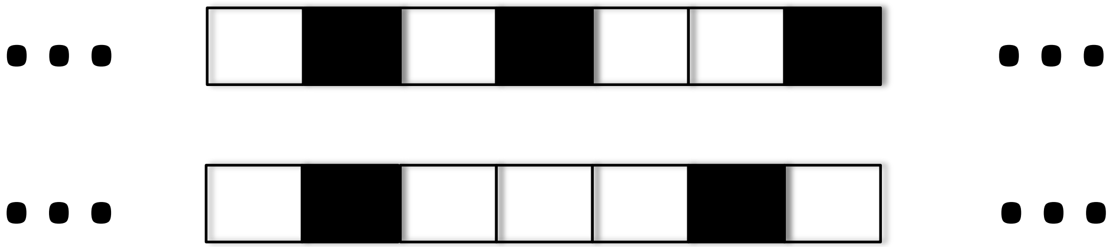
Given $Ax = b$ and $Ax' = b'$, do x and x' have intersection support?

A PAIR-WISE TEST

Given Ax = b and Ax’ = b’, do x and x’ have intersection support?

supp(x) =

supp(x’) =

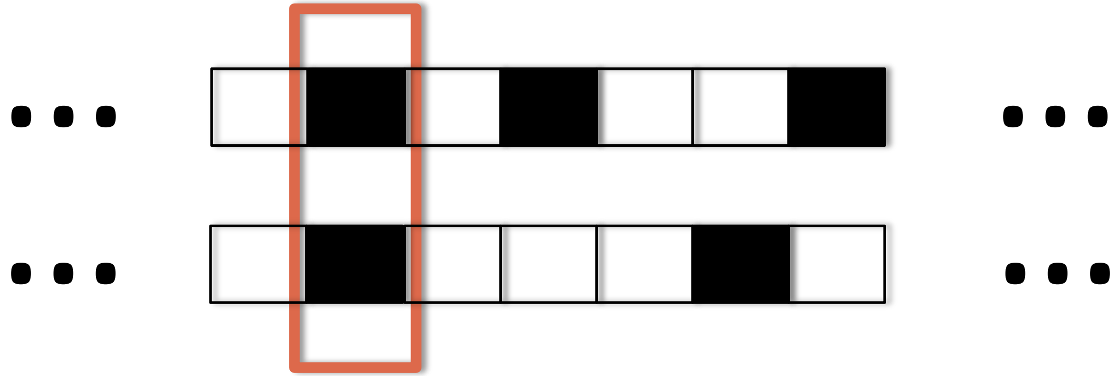


A PAIR-WISE TEST

Given Ax = b and Ax’ = b’, do x and x’ have intersection support?

supp(x) =

supp(x’) =

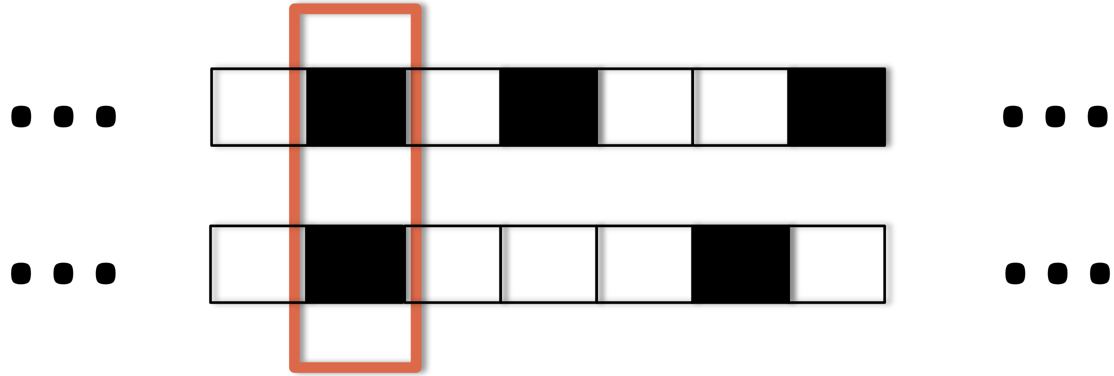


A PAIR-WISE TEST

Given Ax = b and Ax’ = b’, do x and x’ have intersection support?

supp(x) =

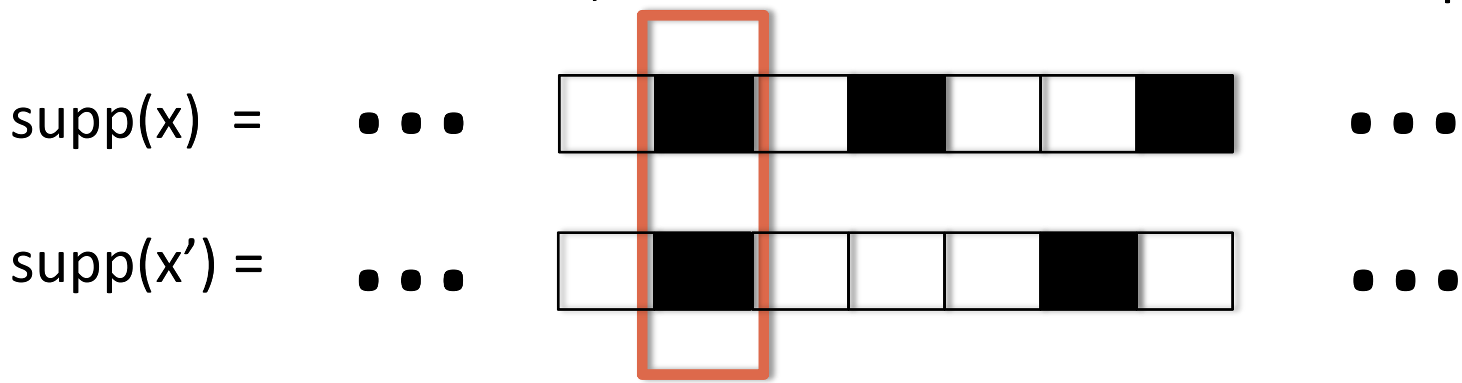
supp(x’) =



$\langle x', x \rangle$ $\begin{cases} \text{zero} & \text{maybe} \\ \text{non-zero} & \text{yes} \end{cases}$

A PAIR-WISE TEST

Given Ax = b and Ax’ = b’, do x and x’ have intersection support?



$\langle x', x \rangle$: zero → maybe; non-zero → yes

$$\langle x', x \rangle \approx \langle Ax', Ax \rangle$$

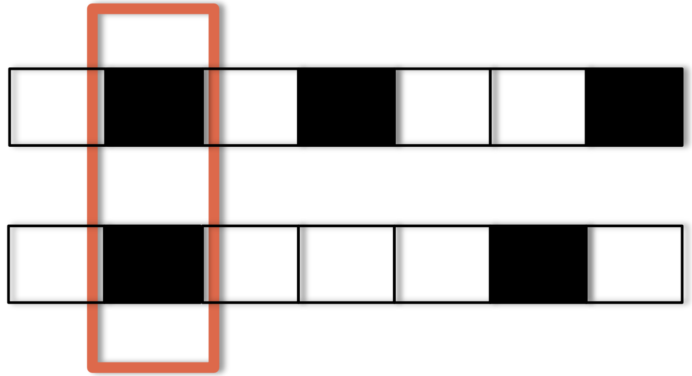
Uncertainty Principle: for k-sparse x, incoherent A

A PAIR-WISE TEST

Given Ax = b and Ax' = b', do x and x' have intersection support?

supp(x) = ...

supp(x') = ...



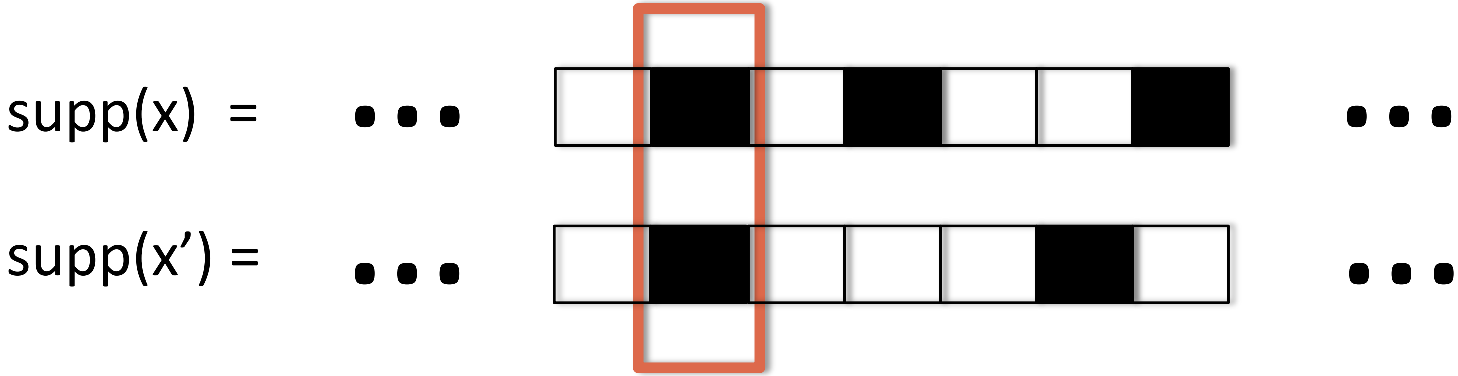
$\langle x', x \rangle$ { zero: maybe; non-zero: yes }

$\langle x', x \rangle \approx \langle Ax', Ax \rangle$ { zero: maybe; non-zero: yes, whp }

Uncertainty Principle: for k-sparse x, incoherent A

A PAIR-WISE TEST

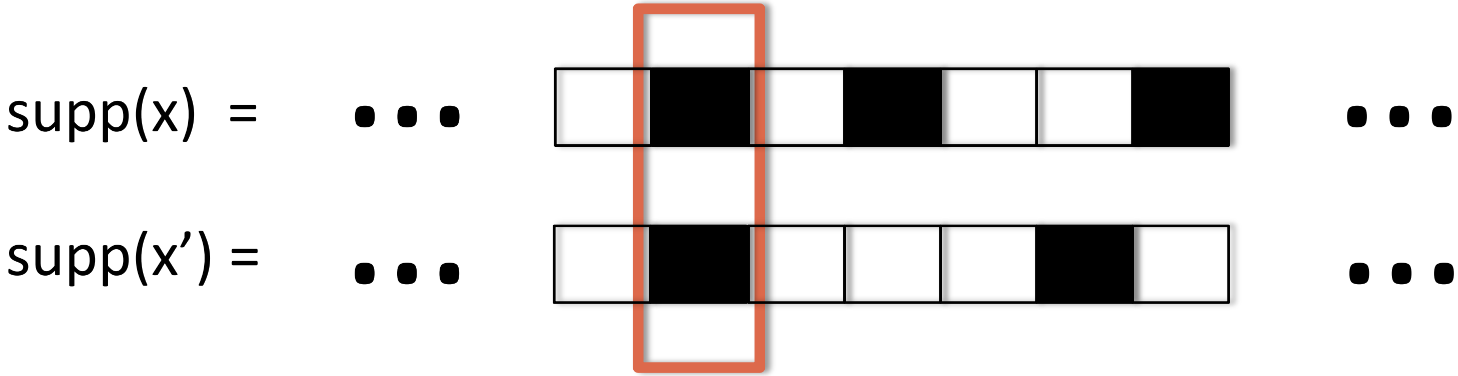
Given $Ax = b$ and $Ax' = b'$, do x and x' have intersection support?



Approach: Build a graph G on the p samples, with an edge btwn b and b' if and only if $|b^T b'| > 1/2$

A PAIR-WISE TEST

Given Ax = b and Ax' = b', do x and x' have intersection support?



Approach: Build a graph G on the p samples, with an edge btwn b and b' if and only if $|b^T b'| > 1/2$

For the purposes of this talk, probability of an edge between b, b' is ½ iff supp(x) and supp(x') intersect

OVERLAPPING CLUSTERING

Let $C_i = \{ b \mid x_i \neq 0 \}$ **(overlapping)**

OVERLAPPING CLUSTERING

Let C_i ={ b | $x_i \neq 0$}   **(overlapping)**

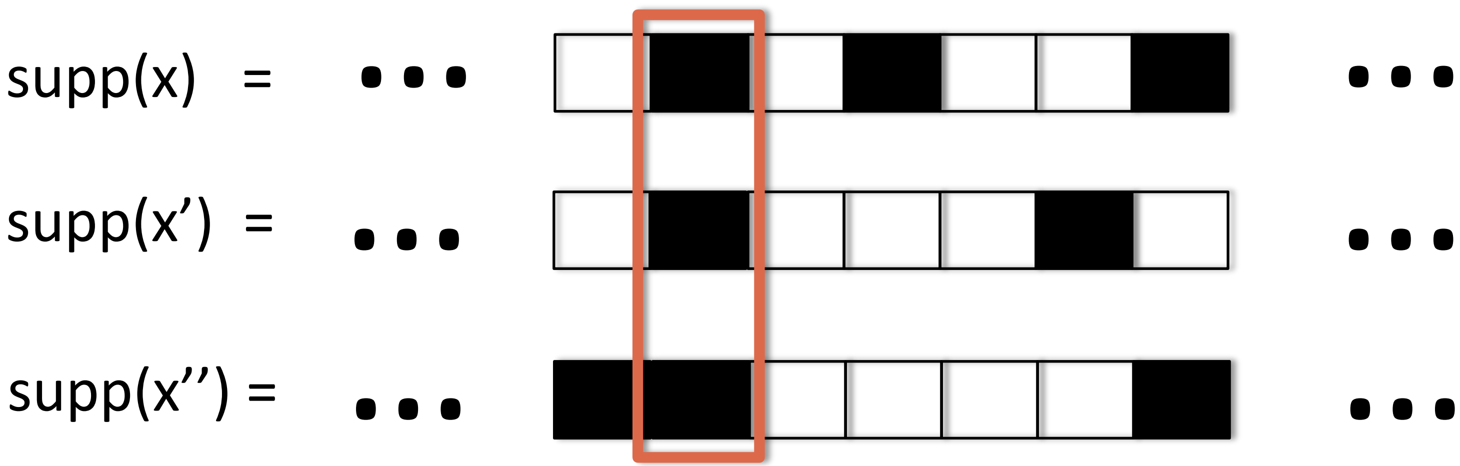
Can we find the clusters efficiently?

OVERLAPPING CLUSTERING

Let C_i ={ b | $x_i \neq 0$} **(overlapping)**

Can we find the clusters efficiently?

Challenge: Given (x, x', x'') where **all the pairs** belong to a cluster together, do all three belong to a common cluster too?



OVERLAPPING CLUSTERING

Let C_i ={ b | $x_i \neq 0$}   **(overlapping)**

Can we find the clusters efficiently?

Challenge: Given (x, x', x'') where **all the pairs** belong to a cluster together, do all three belong to a common cluster too?

OVERLAPPING CLUSTERING

Let $C_i = \{ b \mid x_i \neq 0\}$ **(overlapping)**

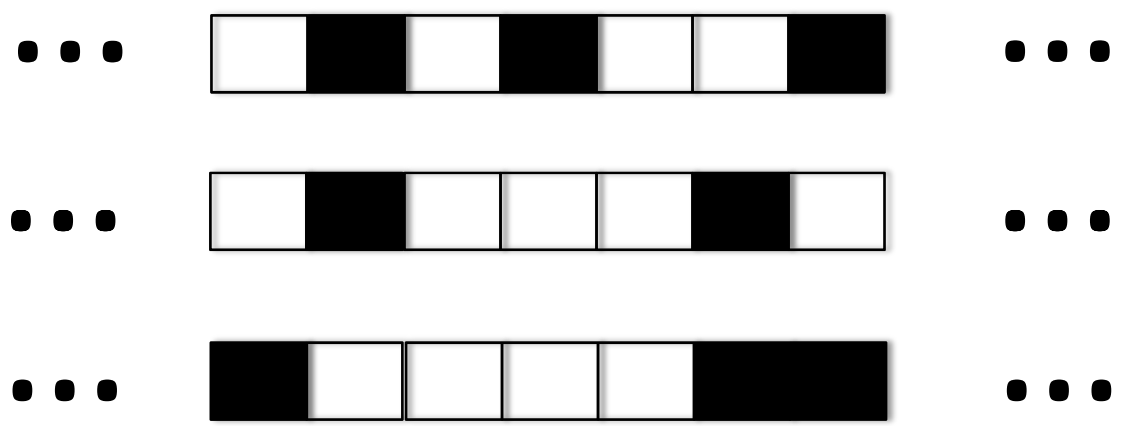
Can we find the clusters efficiently?

Challenge: Given (x, x', x'') where **all the pairs** belong to a cluster together, do all three belong to a common cluster too?

supp(x) =

supp(x') =

supp(x'') =



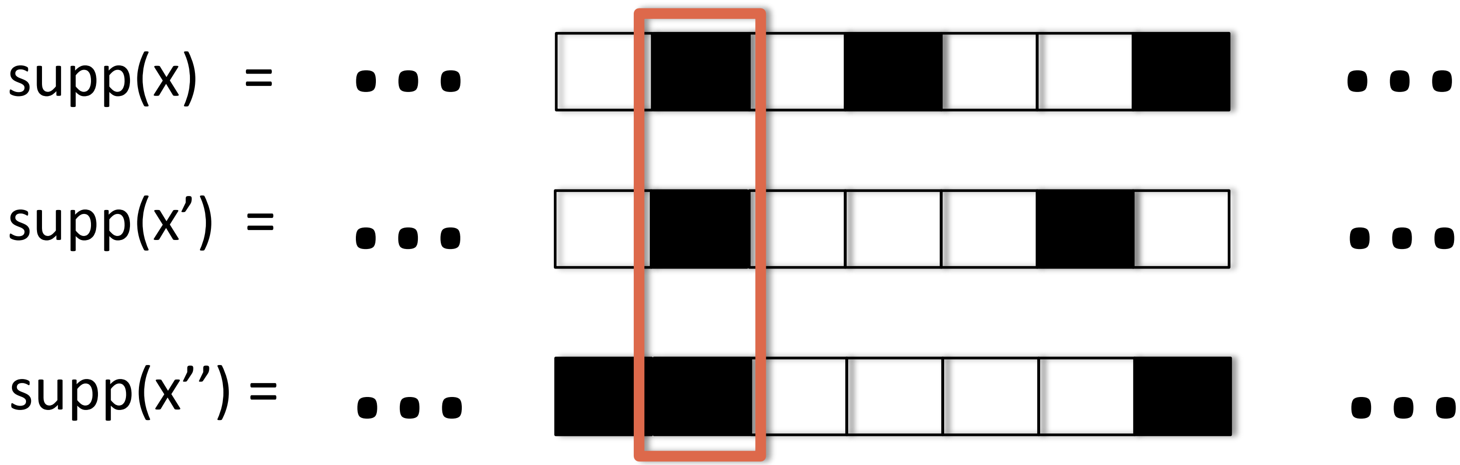
A TRIPLE TEST

Key Idea: Use new samples y …

A TRIPLE TEST

Key Idea: Use new samples y …

Case #1: all three intersect:

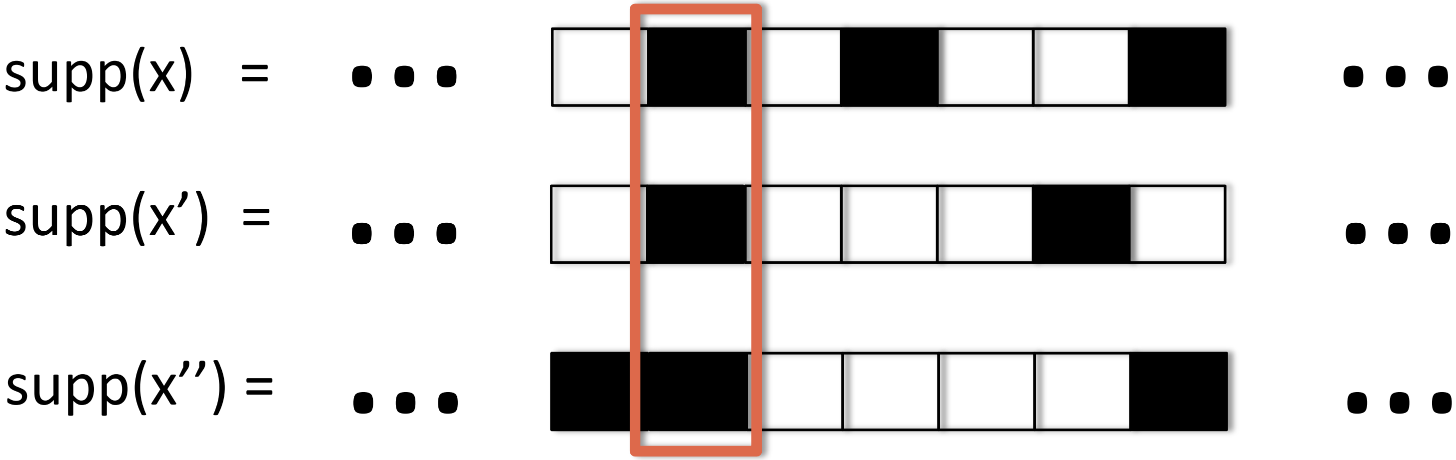


A TRIPLE TEST

Key Idea: Use new samples y …

Case #1: all three intersect:

Probability y intersects all three is at least k/m



New sample y only needs to contain one element from their joint union

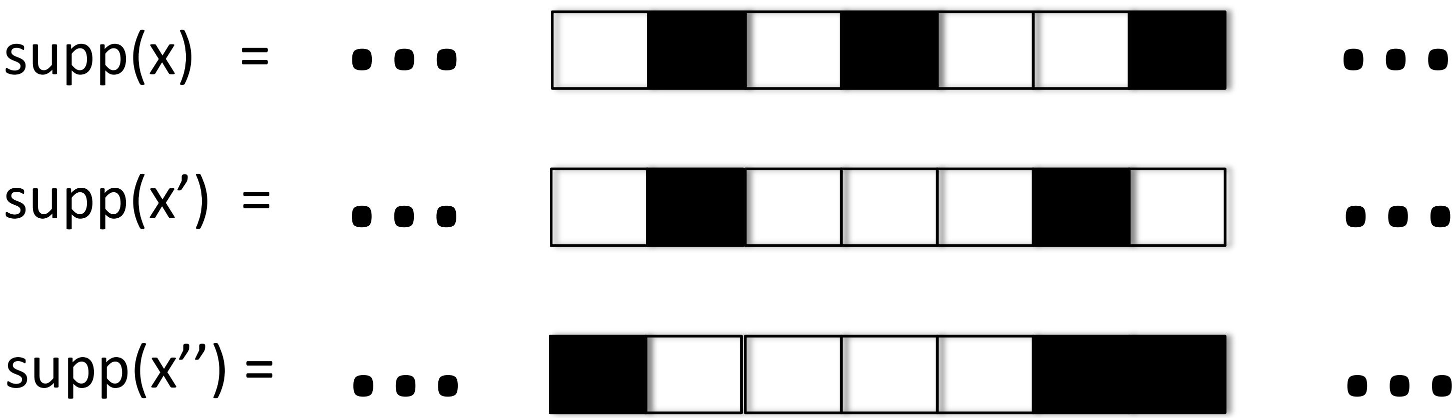
A TRIPLE TEST

Key Idea: Use new samples y …

A TRIPLE TEST

Key Idea: Use new samples y …

Case #2: no common intersection



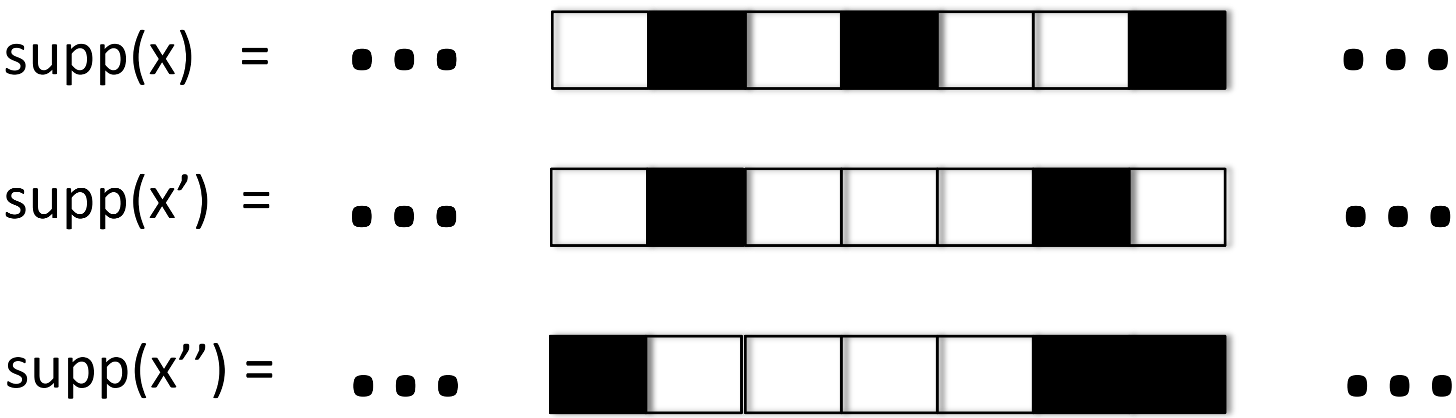
New sample y needs to contain at least two elements from their joint union

A TRIPLE TEST

Key Idea: Use new samples y …

Case #2: no common intersection, $|supp(x) \cap supp(x')| \leq C$, etc

Probability y intersects all three is at most $O(Ck^3/m^2)$



New sample y needs to contain at least two elements from their joint union

A TRIPLE TEST

Key Idea: Use new samples y' …

Case #1: all three intersect:

Probability y intersects all three is at least k/m

Case #2: no common intersection, $|supp(x) \cap supp(x')| \leq C$, etc

Probability y intersects all three is at most $O(Ck^3/m^2)$

A TRIPLE TEST

Key Idea: Use new samples y' …

Case #1: all three intersect:

Probability y intersects all three is at least k/m

Case #2: no common intersection, $|supp(x) \cap supp(x')| \leq C$, etc

Probability y intersects all three is at most $O(Ck^3/m^2)$

Triple Test:

- Given (x, x', x'') where all the pairs intersect
- If there are at least T samples y where (x, x', x'', y) all pairwise intersect, **ACCEPT** else **REJECT**

FINDING ALL THE CLUSTERS

We can build a clustering algorithm on this primitive:

- For each pair (x, x’), find all x’’ that pass the triple test

FINDING ALL THE CLUSTERS

We can build a clustering algorithm on this primitive:

- For each pair (x, x'), find all x'' that pass the triple test

Claim: This set is the union of clusters corresponding to supp(x)$\cap$supp(x')

FINDING ALL THE CLUSTERS

We can build a clustering algorithm on this primitive:

- For each pair (x, x'), find all x'' that pass the triple test

Claim: This set is the union of clusters corresponding to supp(x) $\cap$ supp(x')

Claim: For every cluster i, there is some x, x' that uniquely identify it – i.e. supp(x) $\cap$ supp(x') = {i}

FINDING ALL THE CLUSTERS

We can build a clustering algorithm on this primitive:

- For each pair (x, x'), find all x'' that pass the triple test

Claim: This set is the union of clusters corresponding to supp(x) $\cap$ supp(x')

Claim: For every cluster i, there is some x, x' that uniquely identify it – i.e. supp(x) $\cap$ supp(x') = {i}

- Output inclusion-wise minimal sets – these are the clusters!

FINDING ALL THE CLUSTERS

We can build a clustering algorithm on this primitive:

- For each pair (x, x'), find all x'' that pass the triple test

Claim: This set is the union of clusters corresponding to supp(x) $\cap$ supp(x')

Claim: For every cluster i, there is some x, x' that uniquely identify it – i.e. supp(x) $\cap$ supp(x') = {i}

- Output inclusion-wise minimal sets – these are the clusters!

Our full algorithm uses higher-order tests; analysis through connections to piercing number

Many ways to get the **dictionary** from the **clustering**…

Many ways to get the **dictionary** from the **clustering**...

Approach #1: Relative Signs

Plan: Refine C_i and find all the b's with $x_i > 0$

Many ways to get the **dictionary** from the **clustering**…

Approach #1: Relative Signs

Plan: Refine C_i and find all the b's with $x_i > 0$

Intuition: If $\text{supp}(x) \cap \text{supp}(x') = \{i\}$, the we can find relative sign of x_i and x'_i and there are many such pairs…

Many ways to get the **dictionary** from the **clustering**…

Approach #1: Relative Signs

Plan: Refine C_i and find all the b's with $x_i > 0$

Intuition: If $\mathrm{supp}(x) \cap \mathrm{supp}(x') = \{i\}$, the we can find relative sign of x_i and x'_i and there are many such pairs…

…enough so that whp we can find all relative signs by **transitivity**

Many ways to get the **dictionary** from the **clustering**...

Approach #1: Relative Signs

Plan: Refine C_i and find all the b's with $x_i > 0$

Intuition: If $supp(x) \cap supp(x') = \{i\}$, the we can find relative sign of x_i and x'_i and there are many such pairs...

...enough so that whp we can find all relative signs by **transitivity**

Claim: $\mathbf{E}[b \mid Ax = b \text{ and } x_i > 0] = A_i \, \mathbf{E}[x_i \mid x_i > 0]$

Hence their empirical average converges to A_i

Many ways to get the **dictionary** from the **clustering**…

Many ways to get the **dictionary** from the **clustering**...

Approach #2: SVD

Suppose we restrict to samples b with $x_i \neq 0$....

Many ways to get the **dictionary** from the **clustering**…

Approach #2: SVD

Suppose we restrict to samples b with $x_i \neq 0$….

Intuition: $\mathbf{E}[bb^T | x_i \neq 0]$ has large variance in direction of A_i

Many ways to get the **dictionary** from the **clustering**…

Approach #2: SVD

Suppose we restrict to samples b with $x_i \neq 0$….

Intuition: $\mathbf{E}[bb^T | x_i \neq 0]$ has large variance in direction of A_i

We also show that alternating minimization works when we're close enough…

(geometric convergence)

OUTLINE

Are there efficient algorithms for dictionary learning?

Introduction

- Origins of Sparse Recovery
- A Stochastic Model; Our Results

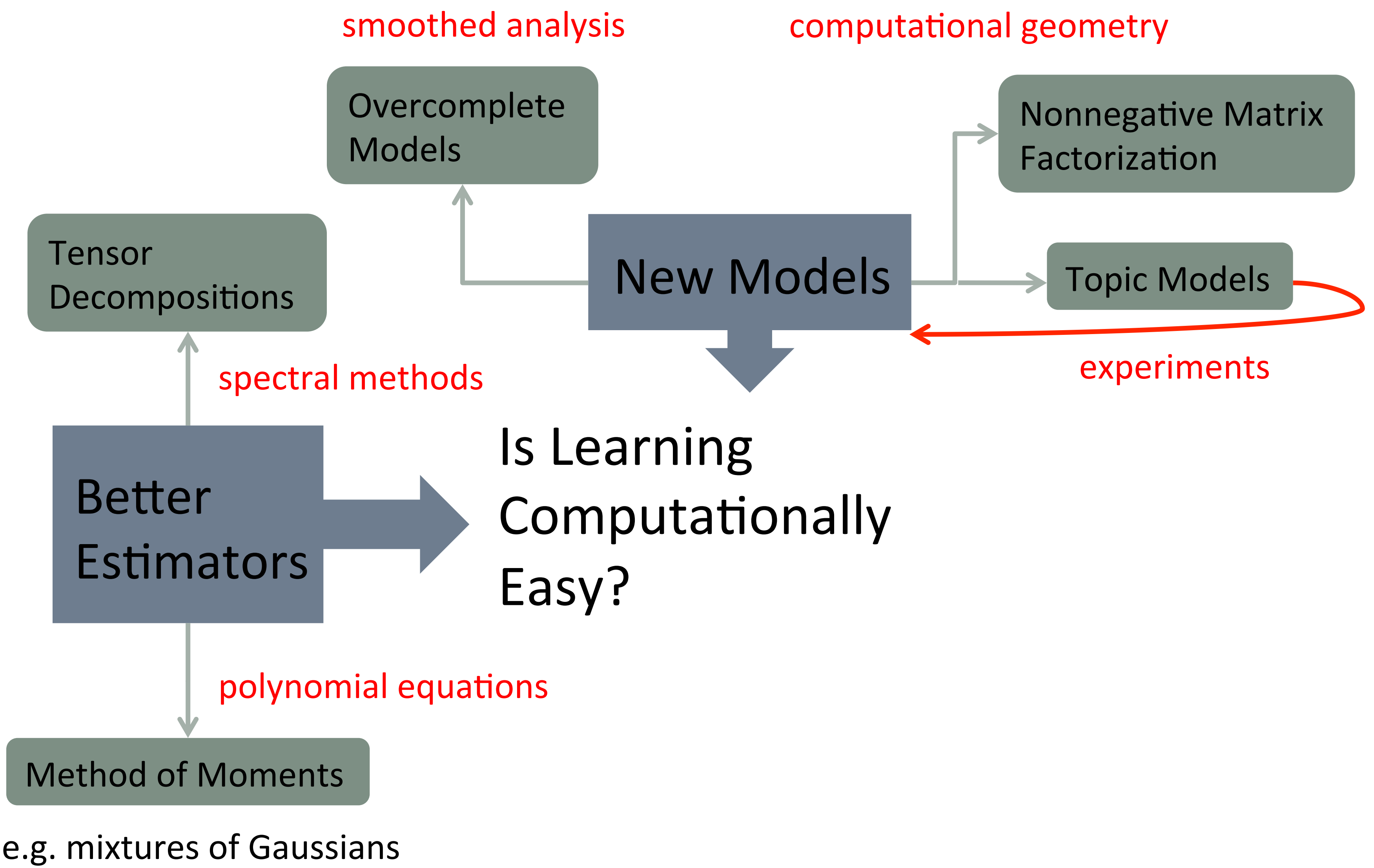
Provable Algorithms via Overlapping Clustering

- Uncertainty Principles
- Reformulation as Overlapping Clustering

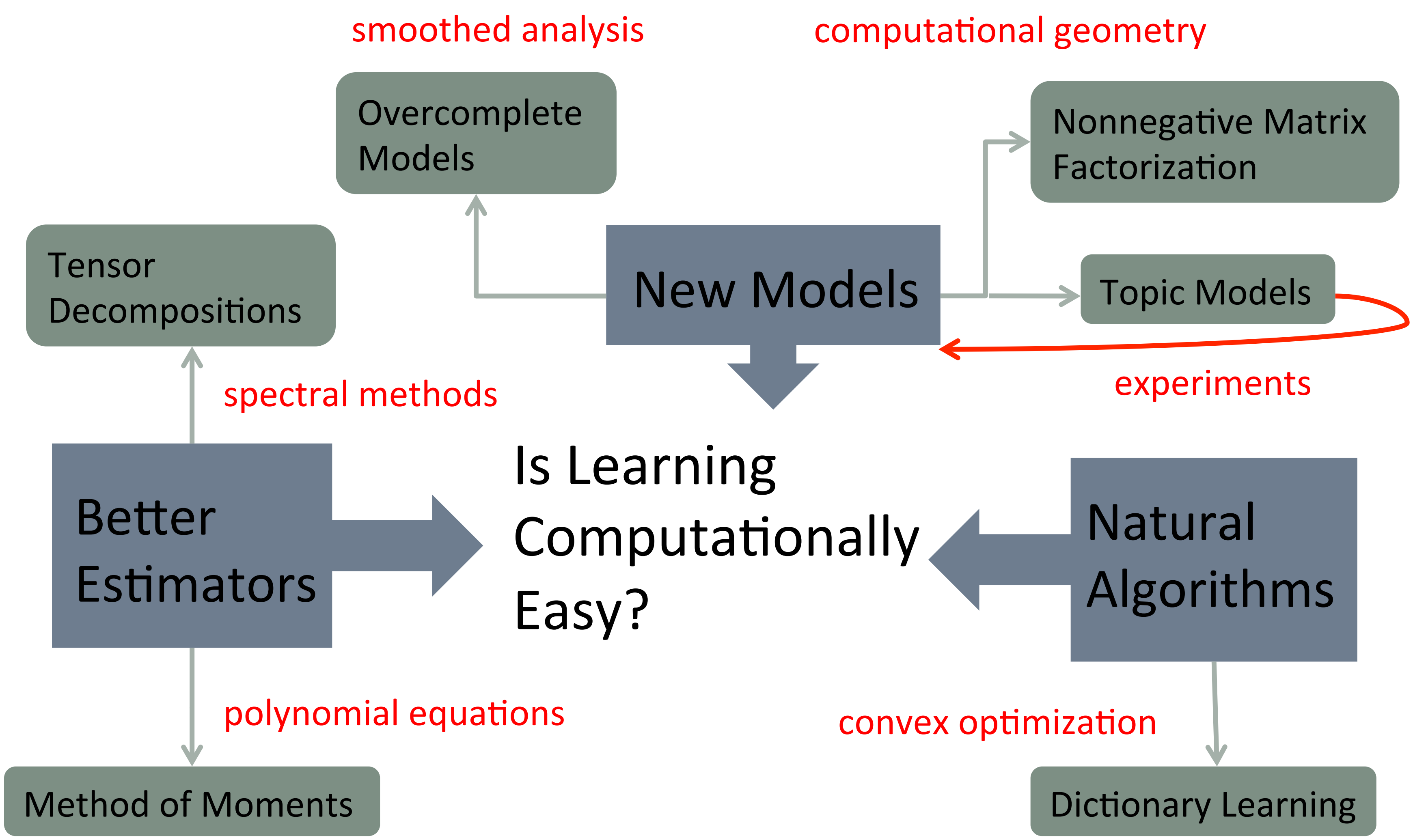
Analyzing Alternating Minimization (out of time)

- Gradient Descent on Non-Convex Fctns

A CONCEPTUAL OVERVIEW



A CONCEPTUAL OVERVIEW



e.g. mixtures of Gaussians

Summary:

- **Provable** algorithms for learning incoherent, overcomplete dictionaries
- Connections to **overlapping** clustering
- Analysis of alternating minimization – gradient descent on non-convex objective
- Why does it work even from a **random initialization**?

Any Questions?

Summary:

- **Provable** algorithms for learning incoherent, overcomplete dictionaries
- Connections to **overlapping** clustering
- Analysis of alternating minimization – gradient descent on non-convex objective
- Why does it work even from a **random initialization**?