Simple, Efficient and Neural Algorithms for Sparse Coding

Ankur Moitra (MIT)

joint work with Sanjeev Arora, Rong Ge and Tengyu Ma
B. A. Olshausen, D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”, 1996
B. A. Olshausen, D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”, 1996

break **natural images** into patches:

(collection of vectors)
B. A. Olshausen, D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”, 1996

break natural images into patches:

(collection of vectors)
B. A. Olshausen, D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”, 1996

break natural images into patches:

(collection of vectors)

Properties: localized, bandpass and oriented
B. A. Olshausen, D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”, 1996

break **natural images** into patches:

(collection of vectors)
B. A. Olshausen, D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”, 1996

break natural images into patches:

(singular value decomposition)

(collection of vectors)
B. A. Olshausen, D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”, 1996

break **natural images** into patches:

Noisy!

Difficult to interpret!

(collection of vectors)
Are there efficient, neural algorithms for sparse coding with **provable guarantees**?
OUTLINE

Are there efficient, neural algorithms for sparse coding with **provable guarantees**?

Part I: The Olshausen-Field Update Rule

- A Non-convex Formulation
- Neural Implementation
- A Generative Model; Prior Work
Are there efficient, neural algorithms for sparse coding with provable guarantees?

Part I: The Olshausen-Field Update Rule

• A Non-convex Formulation
• Neural Implementation
• A Generative Model; Prior Work

Part II: A New Update Rule

• Online, Local and Hebbian with Provable Guarantees
• Connections to Approximate Gradient Descent
• Further Extensions
NONCONVEX FORMULATIONS

Usual approach, minimize reconstruction error:

\[
\min_{A, x^{(i)}'s} \sum_{i=1}^{p} \| b^{(i)} - Ax^{(i)} \| + \sum_{i=1}^{p} L(x^{(i)})
\]

non-linear penalty function
NONCONVEX FORMULATIONS

Usual approach, minimize reconstruction error:

\[
\min_{A, x^{(i)}'s} \sum_{i=1}^{p} \| b^{(i)} - A x^{(i)} \| + \sum_{i=1}^{p} L(x^{(i)})
\]

non-linear penalty function

(encourage sparsity)
NONCONVEX FORMULATIONS

Usual approach, minimize reconstruction error:

\[
\min_{A, \ x^{(i)} \text{'s}} \sum_{i=1}^{p} \| b^{(i)} - A \ x^{(i)} \| + \sum_{i=1}^{p} L(x^{(i)})
\]

non-linear penalty function

(encourage sparsity)

This optimization problem is **NP-hard**, can have many local optima; but **heuristics** work well empirically...
A NEURAL IMPLEMENTATION

[Olshausen, Field]:

image residual output

dictionary stored as synapse weights

residual (stimulus)
A NEURAL IMPLEMENTATION

[Olshausen, Field]:

output

dictionary stored as synapse weights

residual

image (stimulus)

\( b_j \)
A NEURAL IMPLEMENTATION

[Olshausen, Field]:

output

dictionary stored as synapse weights

residual

image (stimulus)
A NEURAL IMPLEMENTATION

[Olshausen, Field]:

Output

Dictionary stored as synapse weights

Residual

Image (stimulus)

\[ \text{image} \]

\[ \text{residual} \]

\[ \text{output} \]
A NEURAL IMPLEMENTATION

[Olshausen, Field]:

output

dictionary stored as synapse weights

residual

image (stimulus)

\[
\begin{align*}
\text{output} & = \text{dictionary} \text{ stored as synapse weights} \\
\text{residual} & = r_j \\
\text{image} (\text{stimulus}) & = b_j \\
\end{align*}
\]
A NEURAL IMPLEMENTATION

[Olshausen, Field]:

Output

Dictionary stored as synapse weights

Residual

Image (stimulus)

\[ L'(x_i) \]
This network performs **gradient descent** by alternating between:

\begin{align}
\text{(1) } & \quad r \leftarrow b - Ax \\
\text{(2) } & \quad x \leftarrow x + \eta (A^T r - \nabla L(x))
\end{align}

And $A$ is updated by a **Hebbian rule**
This network performs **gradient descent** by alternating between:

1. \( r \leftarrow b - Ax \)
2. \( x \leftarrow x + \eta (A^T r - \nabla L(x)) \)

And \( A \) is updated by a Hebbian rule.

Do simple, local and Hebbian rules find **globally** optimal solutions?
This network performs gradient descent by alternating between:

(1) \( r \leftarrow b - Ax \)

(2) \( x \leftarrow x + \eta(A^T r - \nabla L(x)) \)

And A is updated by a Hebbian rule.

Do simple, local and Hebbian rules find globally optimal solutions?

Recent success in analyzing alternating minimization for matrix completion [Jain, Netrapalli, Sanghavi], [Hardt], phase retrieval [Netrapalli, Jain, Sanghavi], robust PCA [Anandkumar et al.], ...
Generative Model:

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

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

[Spielman, Wang, Wright ‘13]: works for full coln rank $A$ up to sparsity roughly $n^{\frac{1}{2}}$ (hence $m \leq n$)
Generative Model:

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

[Spielman, Wang, Wright ‘13]: works for full coln rank $A$ up to sparsity roughly $n^{\frac{1}{2}}$ (hence $m \leq n$)

[Arora, Ge, Moitra ‘14]: works for overcomplete, $\mu$-incoherent $A$ up to sparsity roughly $n^{\frac{1}{2}-\varepsilon}/\mu$
**Generative Model:**

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

---

**[Spielman, Wang, Wright ‘13]:** works for full coln rank $A$ up to sparsity roughly $n^{\frac{1}{2}}$ *(hence $m \leq n$)*

**[Arora, Ge, Moitra ‘14]:** works for overcomplete, $\mu$-incoherent $A$ up to sparsity roughly $n^{\frac{1}{2}-\epsilon}/\mu$

**[Agarwal et al. ‘14]:** works for overcomplete, $\mu$-incoherent $A$ up to sparsity roughly $n^{\frac{1}{4}}/\mu$, via alternating minimization
Generative Model:

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

[Spielman, Wang, Wright ‘13]: works for full coln rank $A$ up to sparsity roughly $n^{\frac{1}{2}}$ (hence $m \leq n$)

[Arora, Ge, Moitra ‘14]: works for overcomplete, $\mu$-incoherent $A$ up to sparsity roughly $n^{\frac{1}{2}-\varepsilon}/\mu$

[Agarwal et al. ‘14]: works for overcomplete, $\mu$-incoherent $A$ up to sparsity roughly $n^{\frac{1}{4}}/\mu$, via alternating minimization

[Barak, Kelner, Steurer ‘14]: works for overcomplete $A$ up to sparsity roughly $n^{1-\varepsilon}$, but running time is exponential in accuracy
OUR RESULTS

Suppose $k \leq \sqrt{n}/\mu \text{ polylog}(n)$ and $\|A\| \leq \sqrt{n} \text{ polylog}(n)$

Suppose $\hat{A}$ that is column-wise $\delta$-close to $A$ for $\delta \leq 1/\text{polylog}(n)$
OUR RESULTS

Suppose $k \leq \sqrt{n}/\mu \text{ polylog}(n)$ and $\|A\| \leq \sqrt{n} \text{ polylog}(n)$

Suppose $\hat{A}$ that is column-wise $\delta$-close to $A$ for $\delta \leq 1/\text{polylog}(n)$

**Theorem [Arora, Ge, Ma, Moitra ‘14]:** There is a neurally plausible update rule that converges to the true dictionary at a geometric rate, and uses a polynomial number of samples
OUR RESULTS

Suppose \( k \leq \sqrt{n}/\mu \) \( \text{polylog}(n) \) and \( \|A\| \leq \sqrt{n} \) \( \text{polylog}(n) \)

Suppose \( \hat{A} \) that is column-wise \( \delta \)-close to \( A \) for \( \delta \leq 1/\text{polylog}(n) \)

Theorem [Arora, Ge, Ma, Moitra ‘14]: There is a neurally plausible update rule that converges to the true dictionary at a geometric rate, and uses a polynomial number of samples.

We also give provable algorithms for initialization based on SVD.
OUR RESULTS

Suppose $k \leq \sqrt{n}/\mu \ \text{polylog}(n)$ and $\|A\| \leq \sqrt{n} \ \text{polylog}(n)$

Suppose $\hat{A}$ that is column-wise $\delta$-close to $A$ for $\delta \leq 1/\text{polylog}(n)$

**Theorem [Arora, Ge, Ma, Moitra ‘14]:** There is a neurally plausible update rule that converges to the true dictionary at a geometric rate, and uses a polynomial number of samples

We also give provable algorithms for initialization based on SVD

---

Our results are based on a new framework for analyzing alternating minimization
Are there efficient, neural algorithms for sparse coding with **provable guarantees**?

**Part I: The Olshausen-Field Update Rule**
- A Non-convex Formulation
- Neural Implementation
- A Generative Model; Prior Work

**Part II: A New Update Rule**
- Online, Local and Hebbian with Provable Guarantees
- Connections to Approximate Gradient Descent
- Further Extensions
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

(1) \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^Tb^{(i)}) \)

(2) \( \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)})\text{sgn}(\hat{x}^{(i)})^T \)
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

1. \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)}) \) (zero out small entries)

2. \( \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)})\text{sgn}(\hat{x}^{(i)})^T \)
A NEW UPDATE RULE

Alternate between the following steps (size $q$ batches):

\[(1) \quad \hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)})\]

\[(2) \quad \hat{A} \leftarrow \hat{A} + \eta \sum_{i = 1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)})\text{sgn}(\hat{x}^{(i)})^T\]
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

(1) $\hat{x}^{(i)} = \text{threshold}(\hat{A}^Tb^{(i)})$

(2) $\hat{A} \leftarrow \hat{A} + \eta \sum_{i = 1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)})\text{sgn}(\hat{x}^{(i)})^T$

The samples arrive online
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

1. \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)}) \)
2. \( \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A} \hat{x}^{(i)}) \text{sgn}(\hat{x}^{(i)})^T \)

The samples arrive online

In contrast, previous (provable) algorithms might need to compute a new estimate from scratch, when new samples arrive
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

(1) \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)}) \)

(2) \( \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)})\text{sgn}(\hat{x}^{(i)})^T \)
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

(1) \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)}) \)

(2) \( \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)}) \text{sgn}(\hat{x}^{(i)})^T \)

The computation is local
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

(1) \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)}) \)

(2) \( \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}x^{(i)}) \text{sgn}(\hat{x}^{(i)})^T \)

The computation is **local**

In particular, the output is a thresholded, weighted sum of activations
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

(1) \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)}) \)

(2) \( \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)})\text{sgn}(\hat{x}^{(i)})^T \)
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

(1) \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)}) \)

(2) \( \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)}) \text{sgn}(\hat{x}^{(i)})^T \)

The update rule is explicitly **Hebbian**
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

1. $\hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)})$

2. $\hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)}) \text{sgn}(\hat{x}^{(i)})^T$

The update rule is explicitly Hebbian

“neurons that fire together, wire together”
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

\[
\begin{align*}
(1) \quad \hat{x}^{(i)} &= \text{threshold}(\hat{A}^T b^{(i)}) \\
(2) \quad \hat{A} &\leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)})\text{sgn}(\hat{x}^{(i)})^T
\end{align*}
\]

The update rule is explicitly Hebbian
A NEW UPDATE RULE

Alternate between the following steps (size q batches):

(1) $\hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)})$

(2) $\hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A} \hat{x}^{(i)}) \text{sgn}(\hat{x}^{(i)})^T$

The update rule is explicitly Hebbian

The update to a weight $\hat{A}_{i,j}$ is the product of the activations at the residual layer and the decoding layer
APPROXIMATE GRADIENT DESCENT

We give a general framework for designing and analyzing iterative algorithms for sparse coding.
APPROXIMATE GRADIENT DESCENT

We give a general framework for designing and analyzing iterative algorithms for sparse coding.

The usual approach is to think of them as trying to minimize a non-convex function:

\[
\min_{\hat{A}, \text{coln-sparse } \hat{X}} E(\hat{A}, \hat{X}) = \| B - \hat{A} \hat{X} \|_F^2
\]
We give a general framework for designing and analyzing iterative algorithms for sparse coding.

The usual approach is to think of them as trying to minimize a non-convex function:

$$\min \ E(\hat{A}, \hat{X}) = \|B - \hat{A}\hat{X}\|_F^2$$

$\hat{A}$, coln-sparse $\hat{X}$

colns are $b^{(i)}$’s

colns are $\hat{x}^{(i)}$’s
APPROXIMATE GRADIENT DESCENT

We give a general framework for designing and analyzing iterative algorithms for sparse coding.

How about thinking of them as trying to minimize an unknown, convex function?
APPROXIMATE GRADIENT DESCENT

We give a general framework for designing and analyzing iterative algorithms for sparse coding.

How about thinking of them as trying to minimize an unknown, convex function?

$$\min_{\hat{A}} E(\hat{A}, X) = \| B - \hat{A} X \|_F^2$$
APPORAXIMATE GRADIENT DESCENT

We give a general framework for designing and analyzing iterative algorithms for sparse coding.

How about thinking of them as trying to minimize an unknown, convex function?

$$\min_{\hat{A}} E(\hat{A}, X) = \| B - \hat{A} X \|_F^2$$

Now the function is strongly convex, and has a global optimum that can be reached by gradient descent!
APPROXIMATE GRADIENT DESCENT

We give a general framework for designing and analyzing iterative algorithms for sparse coding.

How about thinking of them as trying to minimize an unknown, convex function?

\[
\min_{\hat{A}}\ E(\hat{A}, X) = \| B - \hat{A} X \|_F^2
\]

Now the function is strongly convex, and has a global optimum that can be reached by gradient descent!

**New Goal:** Prove that (with high probability) the step (2) is weakly correlated with the gradient.
We give a general framework for designing and analyzing iterative algorithms for sparse coding.

How about thinking of them as trying to minimize an unknown, convex function?
We give a general framework for designing and analyzing iterative algorithms for sparse coding.

How about thinking of them as trying to minimize an unknown, convex function?

[Balakrishnan, Wainwright, Yu] adopt a similar approach to analyze EM, given a suitable initialization.
APPROXIMATE GRADIENT DESCENT

We give a general framework for designing and analyzing iterative algorithms for sparse coding.

How about thinking of them as trying to minimize an unknown, convex function?

[Balakrishnan, Wainwright, Yu] adopt a similar approach to analyze EM, given a suitable initialization.

Their framework is about the local geometry, and ours is about the direction of movement.
CONDITIONS FOR CONVERGENCE
CONDITIONS FOR CONVERGENCE

Consider the following general setup:

- **optimal solution:** $z^*$
- **update:** $z^{s+1} = z^s - \eta g^s$
CONDITIONS FOR CONVERGENCE

Consider the following general setup:

**optimal solution:** \( z^* \)

**update:** \( z^{s+1} = z^s - \eta g^s \)

**Definition:** \( g^s \) is \((\alpha, \beta, \varepsilon_s)\)-correlated with \( z^* \) if for all \( s \):

\[
\langle g^s, z^s - z^* \rangle \geq \alpha \| z^s - z^* \|^2 + \beta \| g^s \|^2 - \varepsilon_s
\]
CONDITIONS FOR CONVERGENCE

Consider the following general setup:

**optimal solution:** \( z^* \)

**update:** \( z^{s+1} = z^s - \eta \ g^s \)

**Definition:** \( g^s \) is \((\alpha, \beta, \varepsilon_s)\)-correlated with \( z^* \) if for all \( s \):

\[
\langle g^s, z^s - z^* \rangle \geq \alpha \| z^s - z^* \|^2 + \beta \| g^s \|^2 - \varepsilon_s
\]

**Theorem:** If \( g^s \) is \((\alpha, \beta, \varepsilon_s)\)-correlated with \( z^* \), and \( \eta \leq 2\beta \) then

\[
\| z^s - z^* \|^2 \leq (1 - 2\alpha\eta)^s \| z^0 - z^* \|^2 + \frac{\max_s \varepsilon_s}{\alpha}
\]
CONDITIONS FOR CONVERGENCE

Consider the following general setup:

**optimal solution:** \( z^* \)

**update:** \( z^{s+1} = z^s - \eta \, g^s \)

**Definition:** \( g^s \) is \((\alpha, \beta, \varepsilon_s)\)-correlated with \( z^* \) if for all \( s \):

\[
\langle g^s, z^s - z^* \rangle \geq \alpha \left\| z^s - z^* \right\|^2 + \beta \left\| g^s \right\|^2 - \varepsilon_s
\]

**Theorem:** If \( g^s \) is \((\alpha, \beta, \varepsilon_s)\)-correlated with \( z^* \), and \( \eta \leq 2\beta \) then

\[
\left\| z^s - z^* \right\|^2 \leq (1-2\alpha\eta)^s \left\| z^0 - z^* \right\|^2 + \frac{\max_s \varepsilon_s}{\alpha}
\]

This follows immediately from the usual proof...
(1) \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^Tb^{(i)}) \)
(1) \( \hat{x}^{(i)} = \text{threshold}(\hat{A}^Tb^{(i)}) \)

**Decoding Lemma:** If \( \hat{A} \) is \( 1/\text{polylog}(n) \)-close to \( A \) and \( \|\hat{A} - A\| \leq 2 \), then decoding recovers the signs correctly (whp)
\[(1) \quad \hat{x}^{(i)} = \text{threshold}(\hat{A}^Tb^{(i)})\]

Decoding Lemma: If \(\hat{A}\) is \(1/\text{polylog}(n)\)-close to \(A\) and \(\|\hat{A} - A\| \leq 2\), then decoding recovers the signs correctly (whp)

\[(2) \quad \hat{A} \leftarrow \hat{A} + \eta \sum_{i = 1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)})\text{sgn}(\hat{x}^{(i)})^T\]
(1) \[ \hat{x}^{(i)} = \text{threshold}(\hat{A}^Tb^{(i)}) \]

Decoding Lemma: If \( \hat{A} \) is \( 1/\text{polylog}(n) \)-close to \( A \) and \( \|\hat{A} - A\| \leq 2 \), then decoding recovers the signs correctly (whp)

(2) \[ \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)}) \text{sgn}(\hat{x}^{(i)})^T \]

Key Lemma: Expectation of (the column-wise) update rule is

\[ \hat{A}_j \leftarrow \hat{A}_j + \xi (I - \hat{A}_j\hat{A}_j^T)A_j + \xi E_R[\hat{A}_R\hat{A}_R^T]A_j + \text{error} \]

where \( R = \text{supp}(x) \setminus j \), if decoding recovers the correct signs
\( \hat{x}^{(i)} = \text{threshold}(\hat{A}^Tb^{(i)}) \)

**Decoding Lemma:** If \( \hat{A} \) is \( 1/\text{polylog}(n) \)-close to \( A \) and \( \|\hat{A} - A\| \leq 2 \), then decoding recovers the signs correctly (w.h.p.

\[ \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)}) \text{sgn}(\hat{x}^{(i)})^T \]

**Key Lemma:** Expectation of (the column-wise) update rule is

\[ \hat{A}_j \leftarrow \hat{A}_j + \xi (I - \hat{A}_j\hat{A}_j^T)A_j + \xi E_R[\hat{A}_R\hat{A}_R^T]A_j + \text{error} \]

where \( R = \text{supp}(x) \setminus j \), if decoding recovers the correct signs
\( (1) \quad \hat{x}^{(i)} = \text{threshold}(\hat{A}^T b^{(i)}) \)

**Decoding Lemma:** If \( \hat{A} \) is \( 1/\text{polylog}(n) \)-close to \( A \) and \( \|\hat{A} - A\| \leq 2 \), then decoding recovers the signs correctly (whp)

\( (2) \quad \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)}) \text{sgn}(\hat{x}^{(i)})^T \)

**Key Lemma:** Expectation of (the column-wise) update rule is

\[
\hat{A}_j \leftarrow \hat{A}_j + \xi (I - \hat{A}_j \hat{A}_j^T) A_j + \xi E_R[\hat{A}_R \hat{A}_R^T] A_j + \text{error}
\]

where \( R = \text{supp}(x) \setminus j \), if decoding recovers the correct signs
\[ \hat{x}^{(i)} = \text{threshold}(\hat{A}^Tb^{(i)}) \]

**Decoding Lemma:** If \( \hat{A} \) is \( 1/\text{polylog}(n) \)-close to \( A \) and \( \|\hat{A} - A\| \leq 2 \), then decoding recovers the signs correctly (whp)

\[ \hat{A} \leftarrow \hat{A} + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{A}\hat{x}^{(i)})\text{sgn}(\hat{x}^{(i)})^T \]

**Key Lemma:** Expectation of (the column-wise) update rule is

\[ \hat{A}_j \leftarrow \hat{A}_j + \xi (I - \hat{A}_j\hat{A}_j^T)A_j + \xi E_R[\hat{A}_R\hat{A}_R^T]A_j + \text{error} \]

where \( R = \text{supp}(x)\setminus j \), if decoding recovers the correct signs

**Auxiliary Lemma:** \( \|\hat{A} - A\| \leq 2 \), remains true throughout if \( \eta \) is small enough and \( q \) is large enough
FURTHER RESULTS

Adjusting an iterative alg. can have subtle effects on its behavior
FURTHER RESULTS

Adjusting an iterative alg. can have subtle effects on its behavior.

We can use our framework to \textit{synthesize} new update rules.
FURTHER RESULTS

Adjusting an iterative alg. can have subtle effects on its behavior

We can use our framework to **synthesize** new update rules

E.g. we can remove the **systemic bias**, by carefully projecting out along the direction being updated
FURTHER RESULTS

Adjusting an iterative alg. can have subtle effects on its behavior.

We can use our framework to **synthesize** new update rules.

E.g. we can remove the **systemic bias**, by carefully projecting out along the direction being updated.

\[
\hat{x}_j^{(i)} = \text{threshold}(\hat{C}_j b^{(i)})
\]

where \( \hat{C}_j = [\text{Proj}_{A_j}(\hat{A}_1), \text{Proj}_{A_j}(\hat{A}_2), \ldots, \hat{A}_j \ldots \text{Proj}_{A_j}(\hat{A}_m)] \)

\[
(2) \quad \hat{A}_j \leftarrow \hat{A}_j + \eta \sum_{i=1}^{q} (b^{(i)} - \hat{C}_j \hat{x}_j^{(i)}) \text{sgn}(\hat{x}_j^{(i)})^T
\]
Any Questions?

Summary:

- **Online, local** and **Hebbian** algorithms for sparse coding that find a globally optimal solution (whp)
- Introduced a framework for analyzing iterative algorithms by thinking of them as trying to minimize an **unknown, convex** function
- The key is working with a generative model
- Is **computational intractability** really a barrier to a rigorous theory of neural computation?
AN INITIALIZATION PROCEDURE

We give an initialization algorithm that outputs $\hat{A}$ that is column-wise $\delta$-close to $A$ for $\delta \leq 1/\text{polylog}(n)$, $\|\hat{A} - A\| \leq 2$
AN INITIALIZATION PROCEDURE

We give an initialization algorithm that outputs $\hat{A}$ that is column-wise $\delta$-close to $A$ for $\delta \leq 1/\text{polylog}(n)$, $\|\hat{A} - A\| \leq 2$

Repeat: (1) Choose samples $b, b'$
AN INITIALIZATION PROCEDURE

We give an initialization algorithm that outputs \( \hat{A} \) that is column-wise \( \delta \)-close to \( A \) for \( \delta \leq 1 / \text{polylog}(n) \), \( \| \hat{A} - A \| \leq 2 \)

Repeat:  

(1) Choose samples \( b, b' \)

(2) Set \( M_{b,b'} = \frac{1}{q} \sum_{i=1}^{q} (b^T b^{(i)}) (b'^T b^{(i)}) b^{(i)} (b^{(i)})^T \)
AN INITIALIZATION PROCEDURE

We give an initialization algorithm that outputs $\hat{A}$ that is column-wise $\delta$-close to $A$ for $\delta \leq 1/\text{polylog}(n)$, $\|\hat{A} - A\| \leq 2$

Repeat: (1) Choose samples $b, b'$

(2) Set $M_{b,b'} = \frac{1}{q} \sum_{i=1}^{q} (b^T b^{(i)}) (b'^T b^{(i)}) b^{(i)} (b^{(i)})^T$

(3) If $\lambda_1(M_{b,b'}) > \frac{k}{m}$ and $\lambda_2 << \frac{k}{m \log m}$

output top eigenvector
AN INITIALIZATION PROCEDURE

We give an initialization algorithm that outputs $\hat{A}$ that is column-wise $\delta$-close to $A$ for $\delta \leq 1/\text{polylog}(n)$, $\|\hat{A} - A\| \leq 2$

Repeat:  
(1) Choose samples $b, b'$

(2) Set $M_{b,b'} = \frac{1}{q} \sum_{i=1}^{q} (b^T b^{(i)}) (b'^T b^{(i)}) b^{(i)} (b^{(i)})^T$

(3) If $\lambda_1(M_{b,b'}) > \frac{k}{m}$ and $\lambda_2 \ll \frac{k}{m \log m}$

output top eigenvector

Key Lemma: If $Ax = b$ and $Ax' = b'$, then condition (3) is satisfied if and only if $\text{supp}(x) \cap \text{supp}(x') = \{j\}$
We give an initialization algorithm that outputs $\hat{A}$ that is column-wise $\delta$-close to $A$ for $\delta \leq 1/\text{polylog}(n)$, $\|\hat{A} - A\| \leq 2$.

Repeat:  

1. Choose samples $b$, $b'$

2. Set $M_{b,b'} = \frac{1}{q} \sum_{i=1}^{q} (b^T b^{(i)}) (b'^T b^{(i)}) b^{(i)} (b^{(i)})^T$

3. If $\lambda_1(M_{b,b'}) > \frac{k}{m}$ and $\lambda_2 \ll \frac{k}{m \log m}$
   
   output top eigenvector

Key Lemma: If $Ax = b$ and $Ax' = b'$, then condition (3) is satisfied if and only if $\text{supp}(x) \cap \text{supp}(x') = \{j\}$ in which case, the top eigenvector is $\delta$-close to $A_j$.
DISCUSSION

Our initialization gets us to $\delta \leq 1/\text{polylog}(n)$, can be neurally implemented with Oja’s Rule
DISCUSSION

Our initialization gets us to $\delta \leq 1/poly\log(n)$, can be neurally implemented with Oja’s Rule.

Earlier analyses of alternating minimization for $\delta \leq 1/poly(n)$ in [Arora, Ge, Moitra ’14] and [Agarwal et al ’14]
DISCUSSION

Our initialization gets us to $\delta \leq 1/\text{polylog}(n)$, can be neurally implemented with Oja’s Rule.

Earlier analyses of alternating minimization for $\delta \leq 1/\text{poly}(n)$ in [Arora, Ge, Moitra ‘14] and [Agarwal et al ’14]

However, in those settings $A$ and $\hat{A}$ are so close that the objective function is essentially convex.
DISCUSSION

Our initialization gets us to $\delta \leq 1/\text{polylog}(n)$, can be neurally implemented with **Oja’s Rule**

Earlier analyses of alternating minimization for $\delta \leq 1/\text{poly}(n)$ in [Arora, Ge, Moitra ‘14] and [Agarwal et al ’14]

However, in those settings $A$ and $\hat{A}$ are so close that the objective function is **essentially convex**

We show that it converges even from **mild** starting conditions
DISCUSSION

Our initialization gets us to $\delta \leq 1/\text{polylog}(n)$, can be neurally implemented with **Oja’s Rule**

Earlier analyses of alternating minimization for $\delta \leq 1/\text{poly}(n)$ in [Arora, Ge, Moitra ‘14] and [Agarwal et al ’14]

However, in those settings $A$ and $\hat{A}$ are so close that the objective function is **essentially convex**

We show that it converges even from **mild** starting conditions

As a result, our bounds improve on existing algorithms in terms of **running time, sample complexity** and **sparsity** (all but SOS)