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

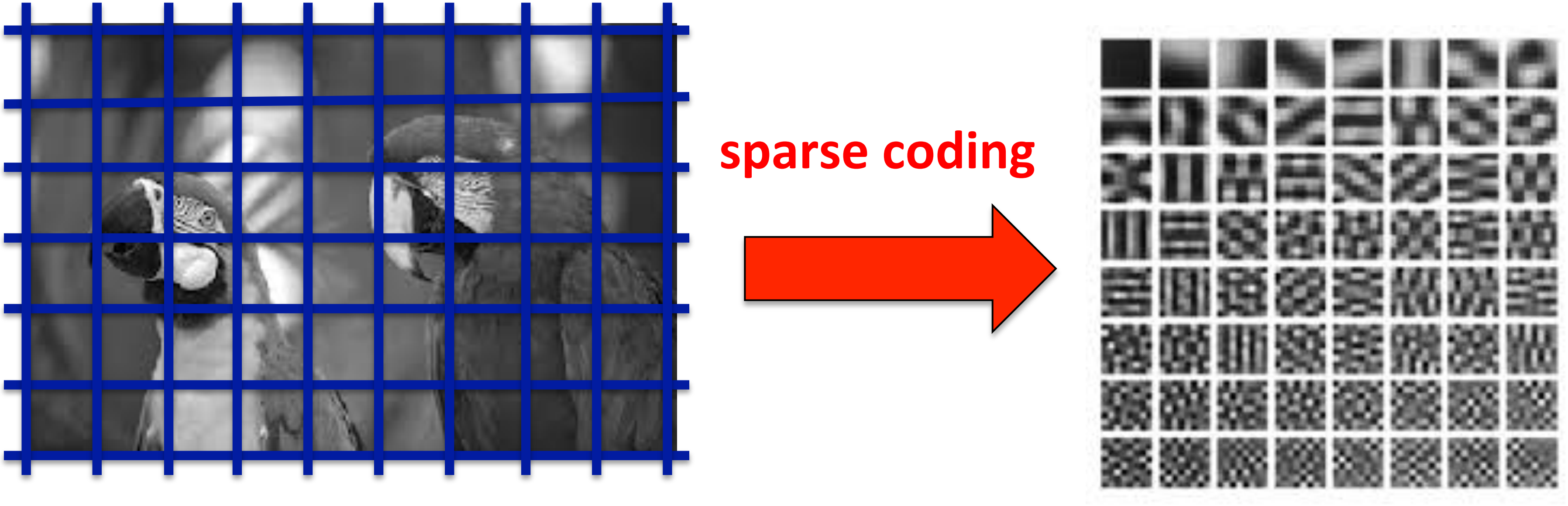
B. A. Olshausen, D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”, 1996

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:



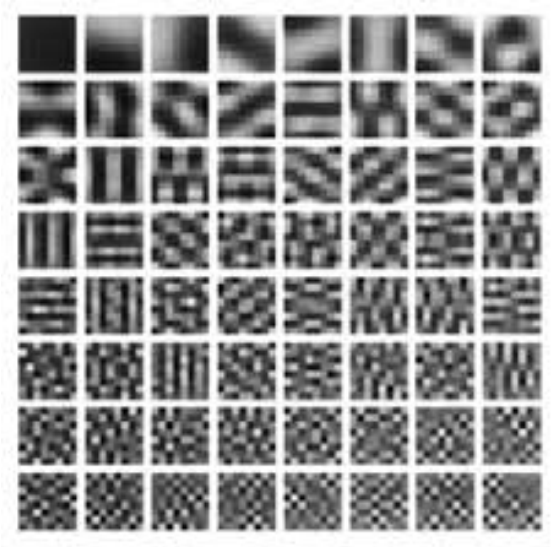
sparse coding

(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:

sparse coding



(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

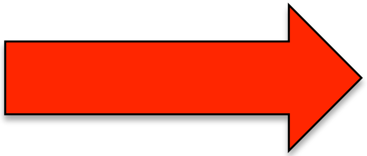
break **natural images** into patches:

(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:

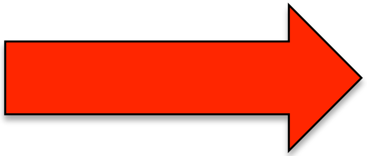
singular value decomposition



(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:



singular value decomposition

Noisy!

Difficult to interpret!

(collection of vectors)

OUTLINE

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

OUTLINE

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

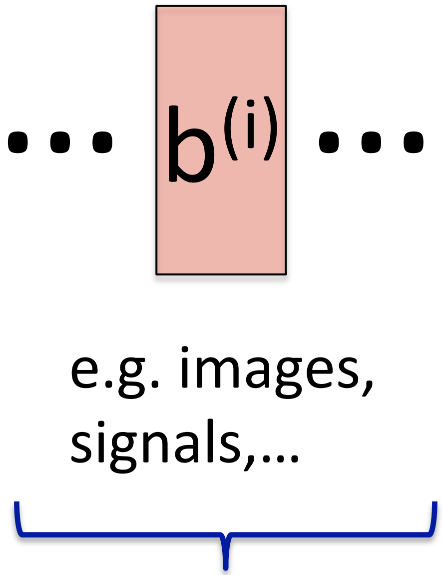
Part I: The Olshausen-Field Update Rule

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

Part II: A New Update Rule

- Online, Local and Hebbian with Provable Guarantees
- Connections to Approximate Gradient Descent
- Further Extensions

More generally, many types of data are sparse in an appropriately chosen basis:

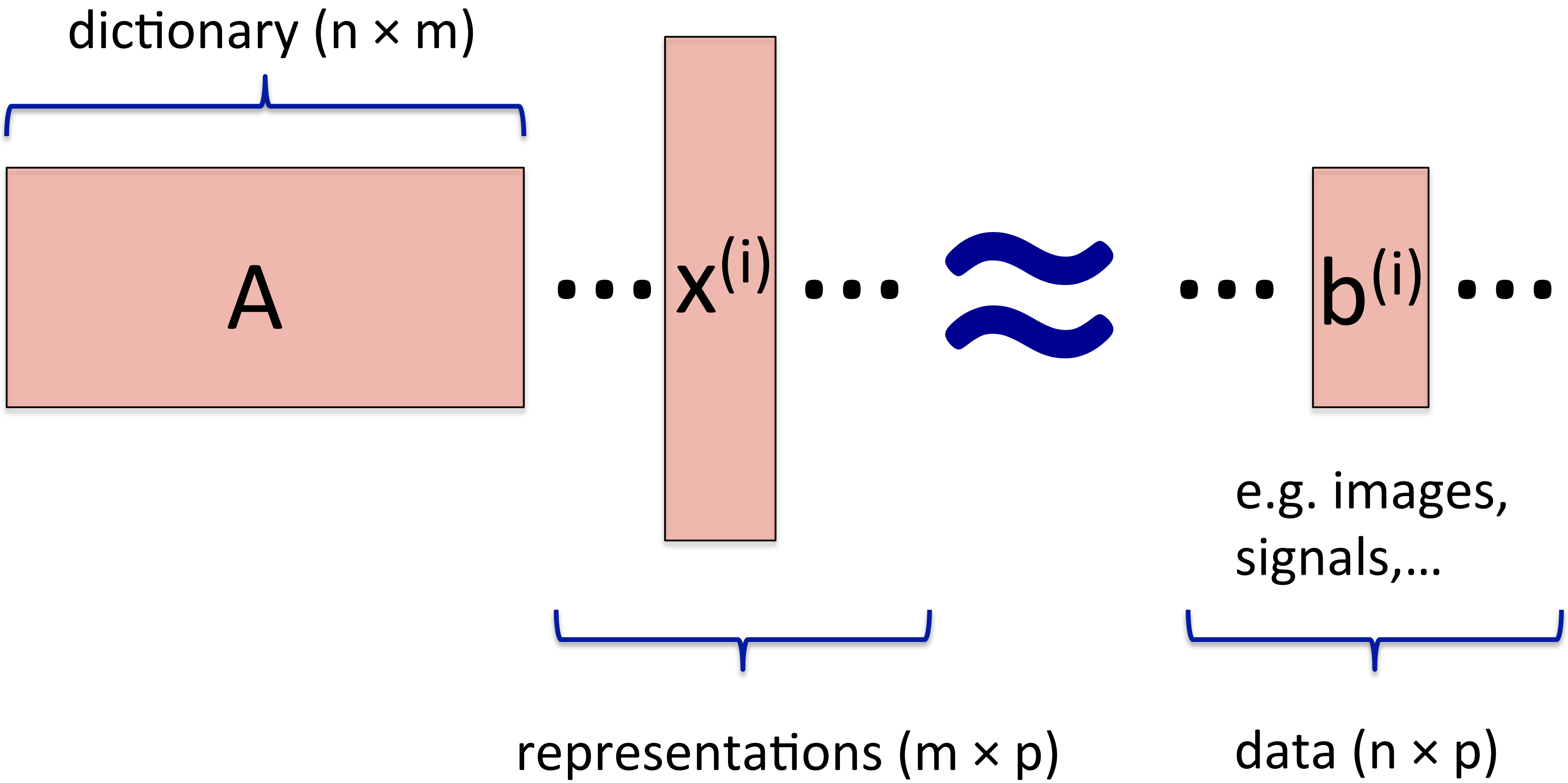


$\cdots \; b^{(i)} \; \cdots$

e.g. images, signals,…

data (n × p)

More generally, many types of data are sparse in an appropriately chosen basis:



dictionary (n × m)

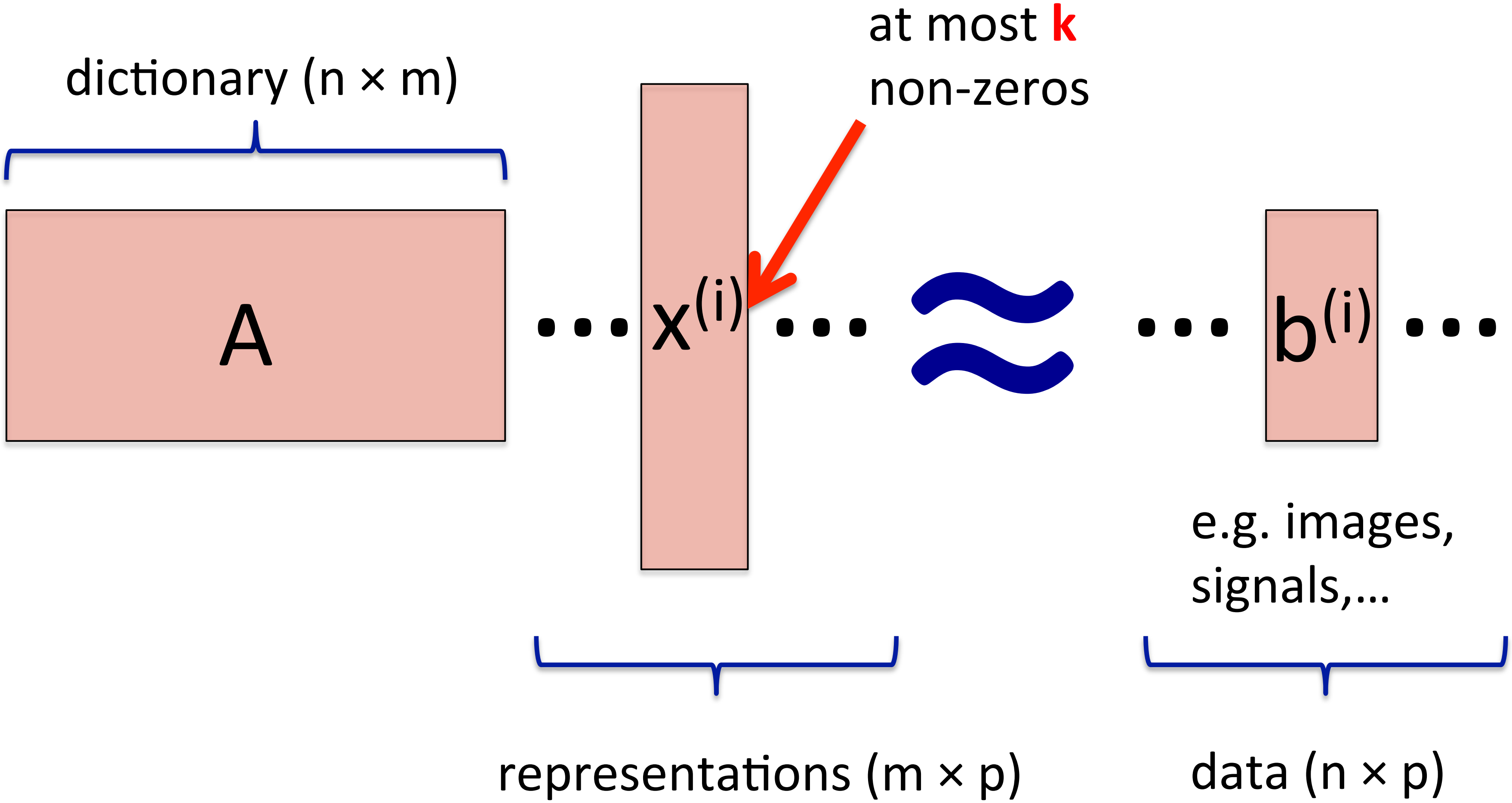
$A \cdots x^{(i)} \cdots \approx \cdots b^{(i)} \cdots$

e.g. images, signals,…

representations (m × p)

data (n × p)

More generally, many types of data are sparse in an appropriately chosen basis:



dictionary (n × m)

at most **k** non-zeros

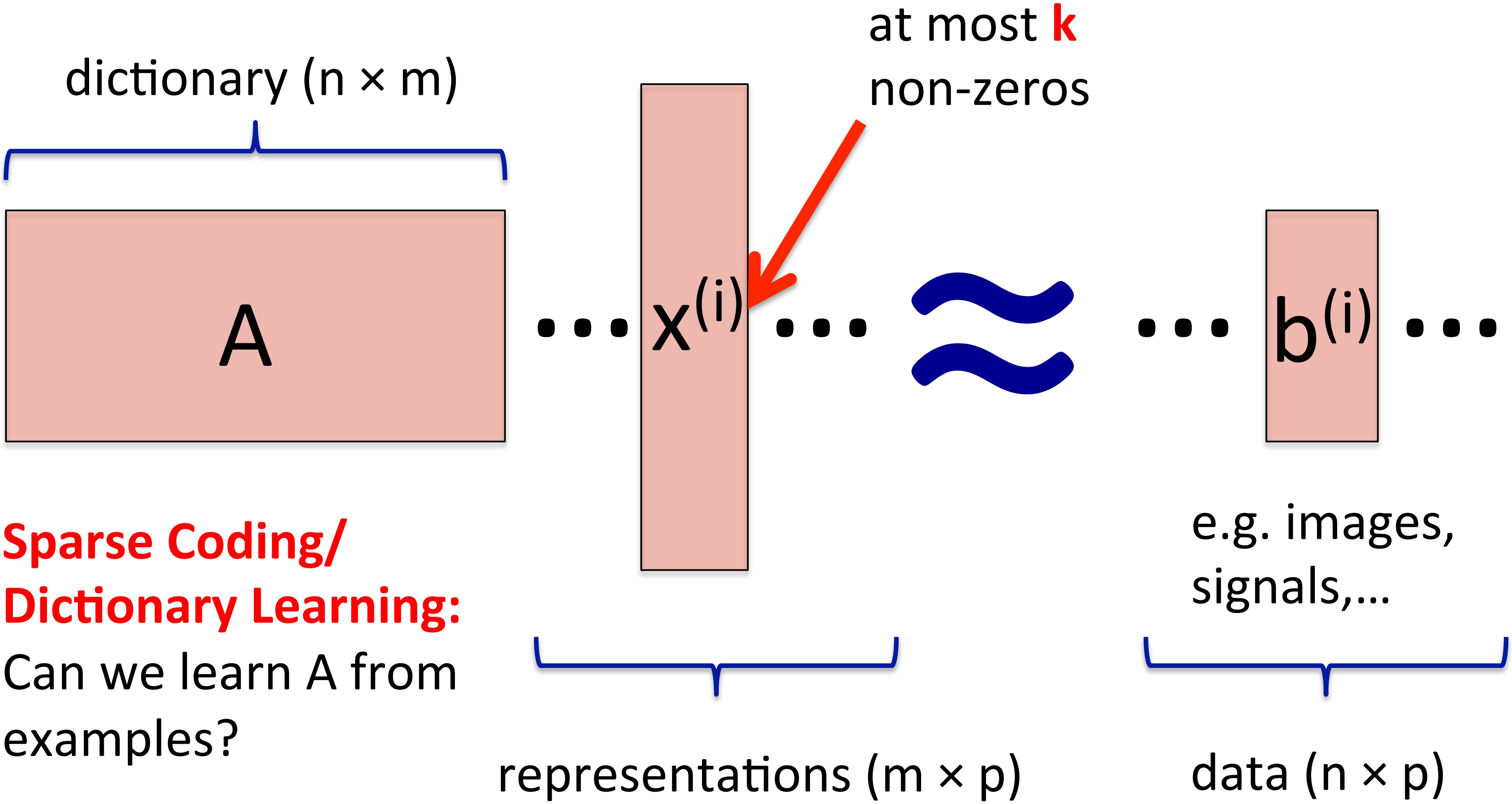
$$A \quad \cdots x^{(i)} \cdots \approx \cdots b^{(i)} \cdots$$

e.g. images, signals,…

representations (m × p)

data (n × p)

More generally, many types of data are sparse in an appropriately chosen basis:



dictionary (n × m)

at most **k** non-zeros

$A \cdots x^{(i)} \cdots \approx \cdots b^{(i)} \cdots$

e.g. images, signals,…

representations (m × p)

data (n × p)

Sparse Coding/ Dictionary Learning: Can we learn A from examples?

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

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)

NONCONVEX FORMULATIONS

Usual approach, minimize **reconstruction error**:

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

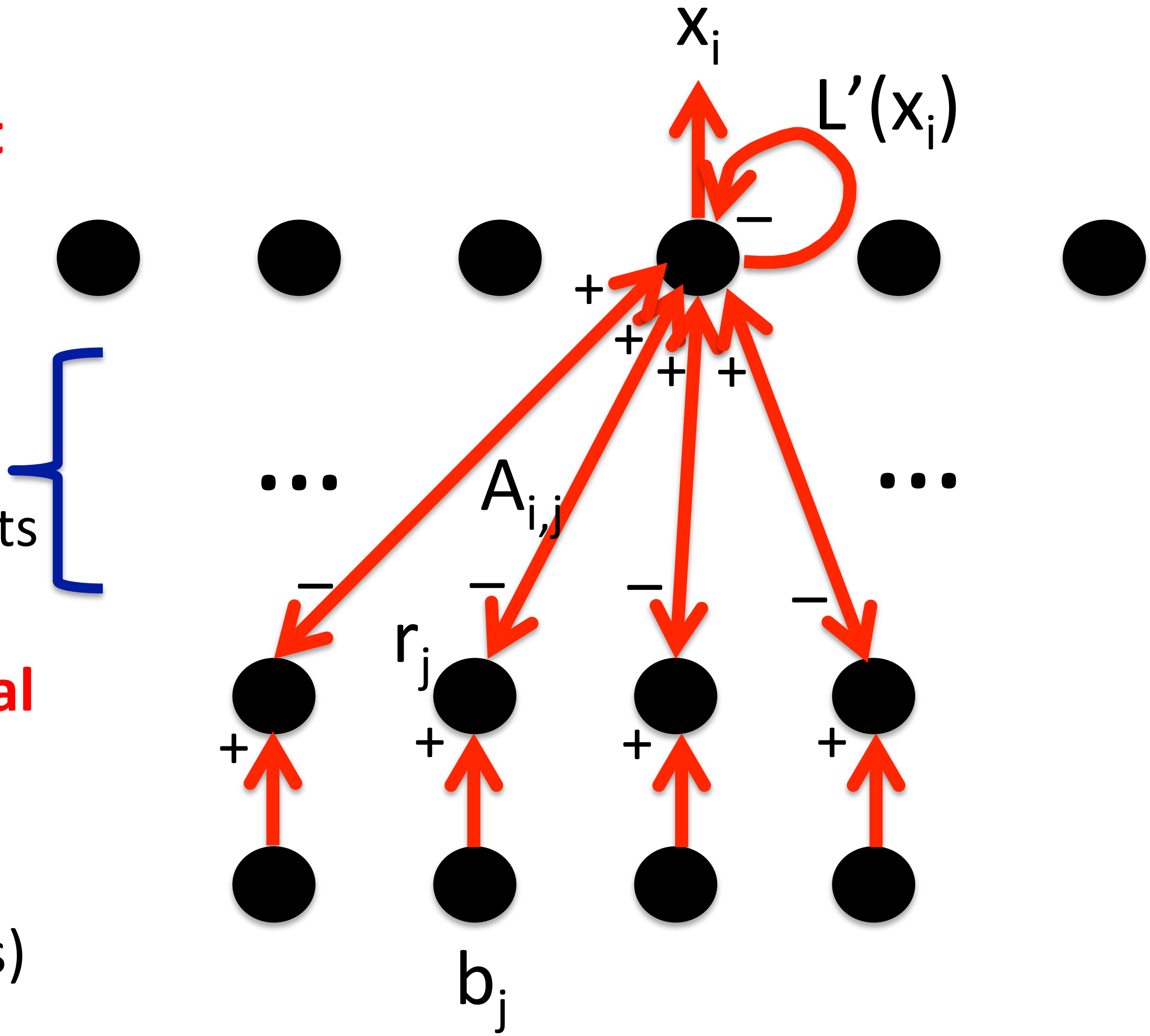
non-linear penalty function

(encourage sparsity)

This optimization problem is **NP-hard**, can have many local optima; but **heuristics** work well nevertheless…

A NEURAL IMPLEMENTATION

[Olshausen, Field]:



output

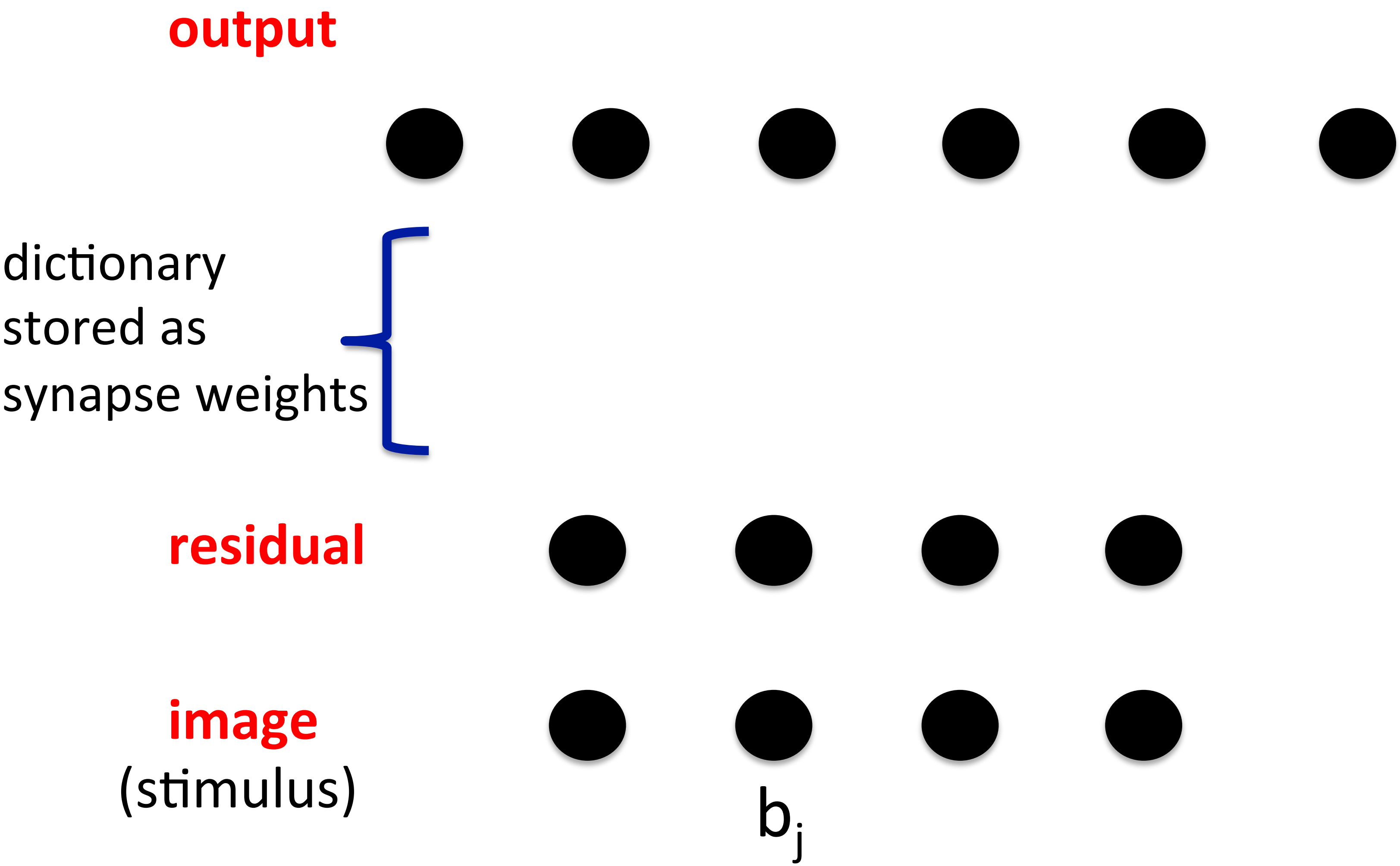
dictionary
stored as
synapse weights

residual

image
(stimulus)

A NEURAL IMPLEMENTATION

[Olshausen, Field]:



output

dictionary
stored as
synapse weights

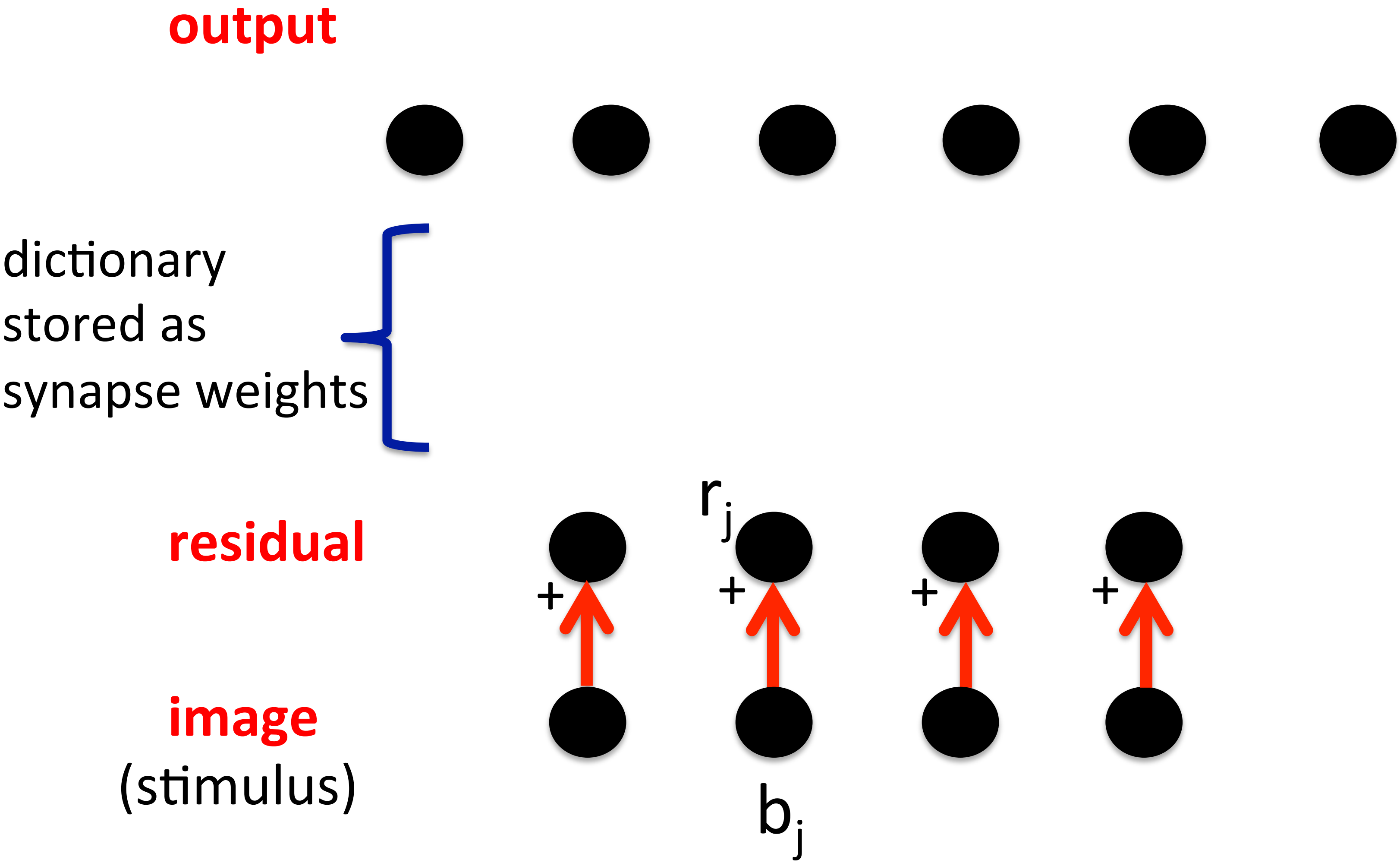
residual

image
(stimulus)

b_j

A NEURAL IMPLEMENTATION

[Olshausen, Field]:



output

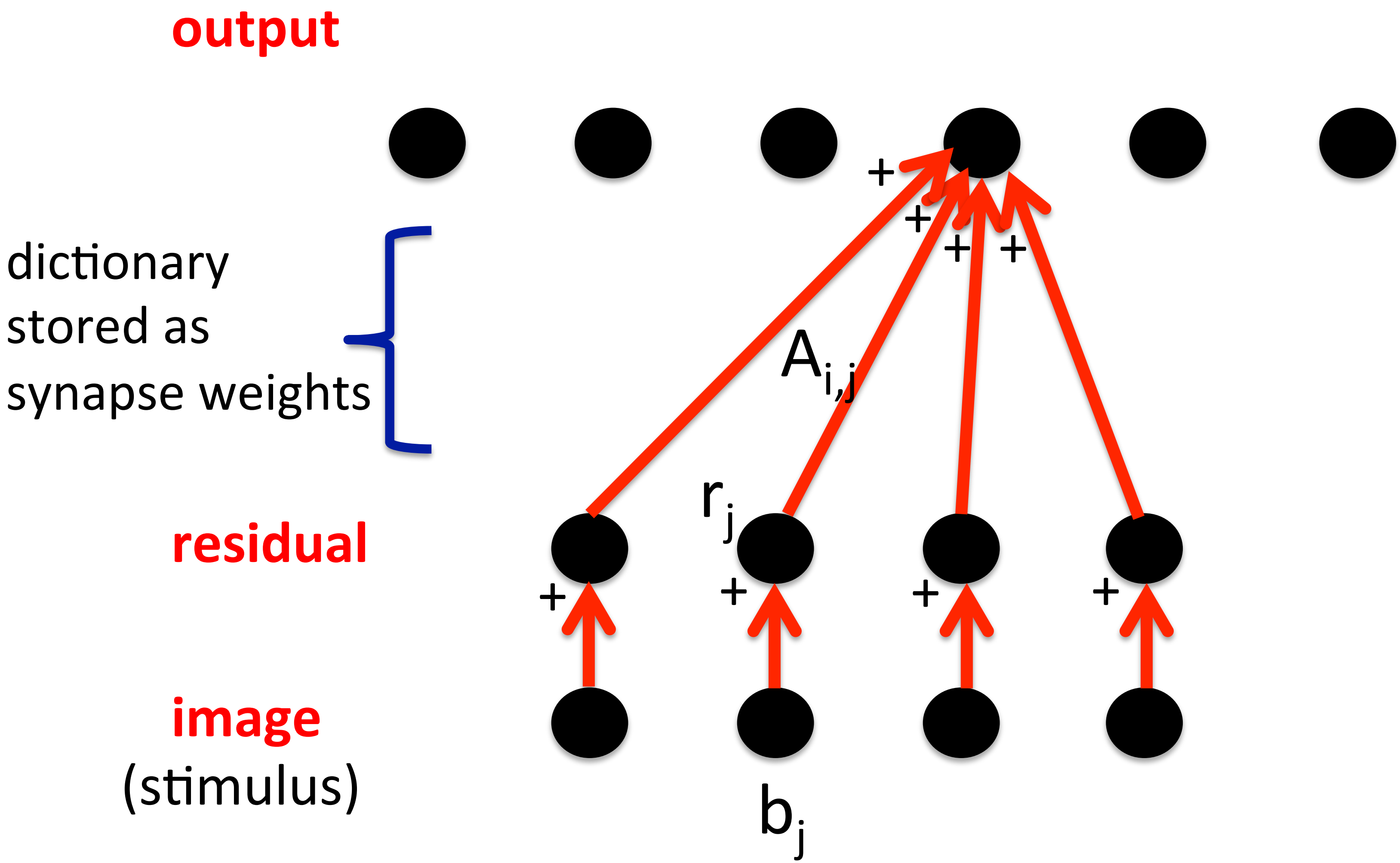
dictionary
stored as
synapse weights

residual

image
(stimulus)

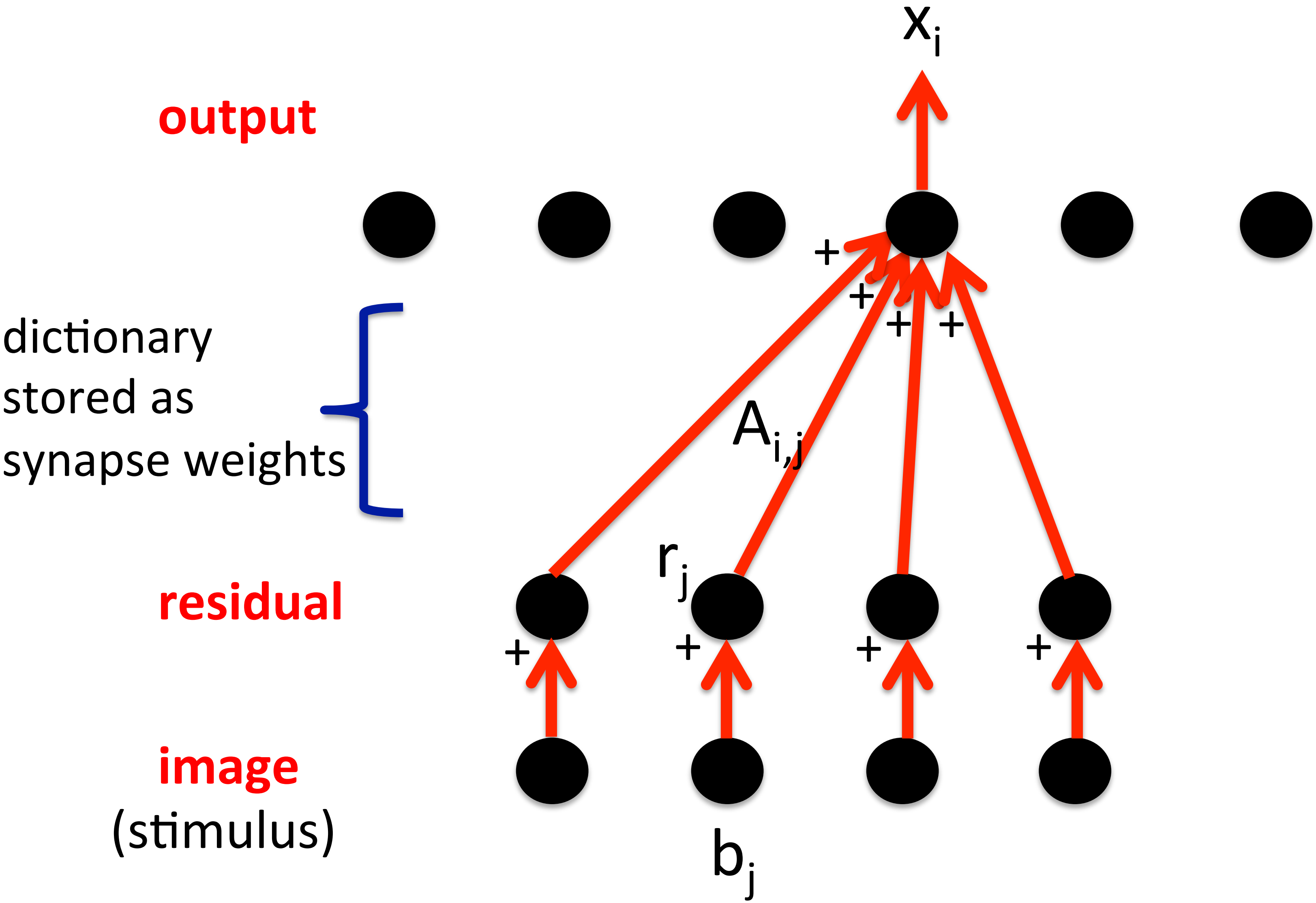
A NEURAL IMPLEMENTATION

[Olshausen, Field]:



A NEURAL IMPLEMENTATION

[Olshausen, Field]:



output

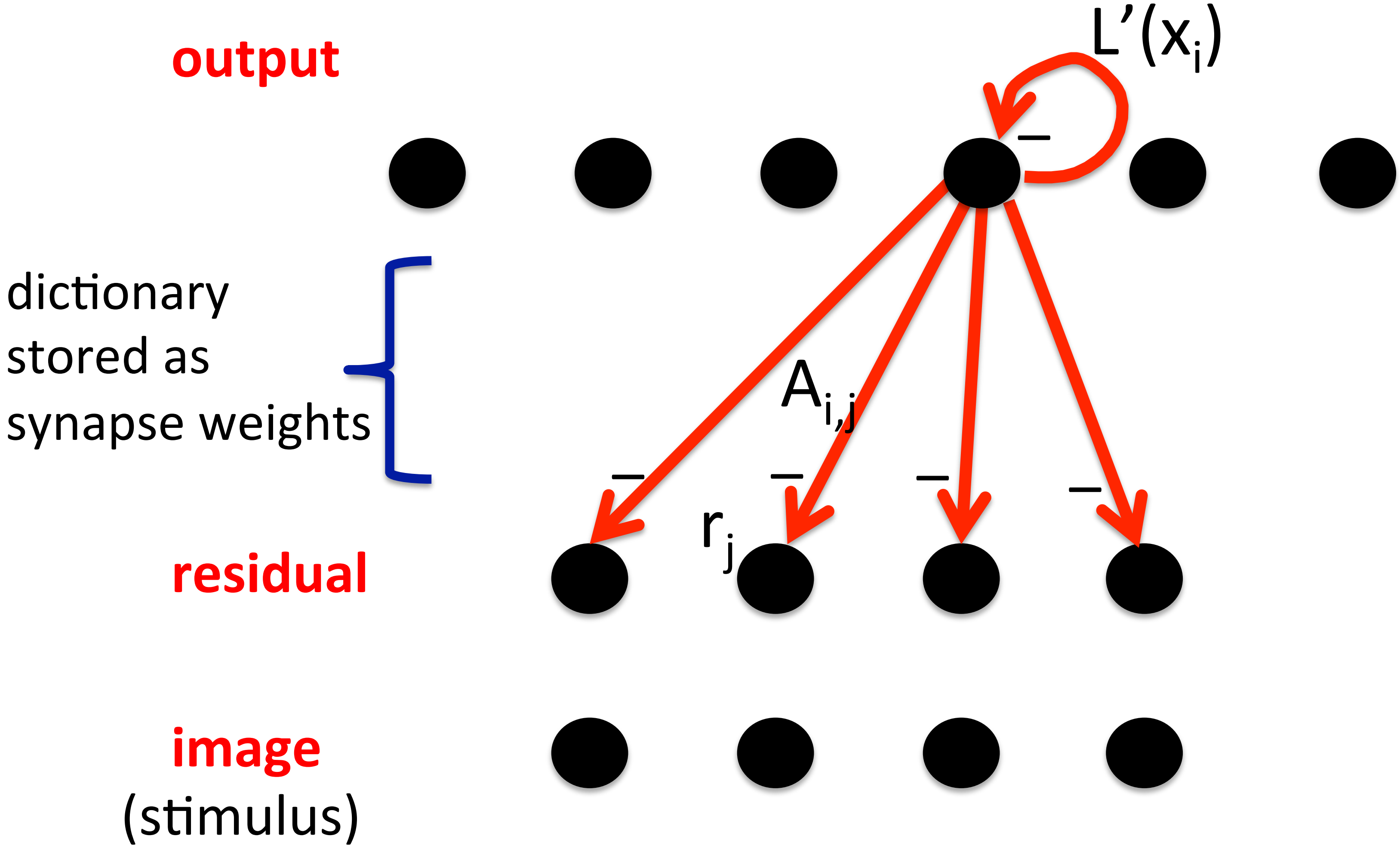
dictionary
stored as
synapse weights

residual

image
(stimulus)

A NEURAL IMPLEMENTATION

[Olshausen, Field]:



This network performs **gradient descent** on:

$$\|b - A\,x\|^2 + L(x)$$

This network performs **gradient descent** on:

$$\|b - A x\|^2 + L(x)$$

by alternating between **(1)** $r \leftarrow b - Ax$

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

This network performs **gradient descent** on:

$$||b - A x||^2 + L(x)$$

by alternating between **(1)** $r \leftarrow b - Ax$

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

Moreover A is updated through **Hebbian rules**

This network performs **gradient descent** on:

$$\|b - A x\|^2 + L(x)$$

by alternating between **(1)** $r \leftarrow b - Ax$

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

Moreover A is updated through **Hebbian rules**

There are no **provable guarantees**, but works well

This network performs **gradient descent** on:

$$||b - A x||^2 + L(x)$$

by alternating between **(1)** $r \leftarrow b - Ax$

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

Moreover A is updated through **Hebbian rules**

There are no **provable guarantees**, but works well

But **why** should gradient descent on a non-convex function work?

This network performs **gradient descent** on:

$$\|b - A\,x\|^2 + L(x)$$

by alternating between **(1)** $r \leftarrow b - Ax$

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

Moreover A is updated through **Hebbian rules**

There are no **provable guarantees**, but works well

But **why** should gradient descent on a non-convex function work?

Are simple, local and Hebbian rules sufficient to find **globally** optimal solutions?

OTHER APPROACHES, AND APPLICATIONS

OTHER APPROACHES, AND APPLICATIONS

Signal Processing/Statistics (MOD, kSVD):

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

OTHER APPROACHES, AND APPLICATIONS

Signal Processing/Statistics (MOD, kSVD):

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

Machine Learning (LBRN06, …):

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

OTHER APPROACHES, AND APPLICATIONS

Signal Processing/Statistics (MOD, kSVD):

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

Machine Learning (LBRN06, …):

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

Theoretical Computer Science (SWW13, AGM14, AAJNT14):

- New algorithms with **provable** guarantees, in a natural **generative model**

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 $n^{1/2}$ **(hence $m \le n$)**

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, μ-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 $n^{1/2}$ **(hence m ≤ n)**

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

[Agarwal et al. '14]: works for overcomplete, μ-incoherent A up to sparsity roughly $n^{1/4}/\mu$, 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, μ-incoherent A up to sparsity roughly $n^{\frac{1}{2}-\varepsilon}/\mu$

[Agarwal et al. ‘14]: works for overcomplete, μ-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 $k \leq \sqrt{n}/\mu$ polylog(n) and $||A|| \leq \sqrt{n}$ polylog(n)

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

OUR RESULTS

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

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

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

OUR RESULTS

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

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

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

All previous algorithms had suboptimal sparsity, worked in less generality, or were **exponential** in a natural parameter

OUR RESULTS

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

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

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

All previous algorithms had suboptimal sparsity, worked in less generality, or were **exponential** in a natural parameter

Note: $k \leq \sqrt{n}/2\mu$ is a barrier, even for sparse recovery

OUR RESULTS

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

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

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

All previous algorithms had suboptimal sparsity, worked in less generality, or were **exponential** in a natural parameter

Note: $k \leq \sqrt{n}/2\mu$ is a barrier, even for sparse recovery

i.e. if $k > \sqrt{n}/2\mu$, then x is not necessarily the sparsest soln to $Ax = b$

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

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

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

(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 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):

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

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

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):

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

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

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**

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

WHAT IS NEURALLY PLAUSIBLE, ANYWAYS?

Our update rule (essentially) inherits a neural implementation from **[Olshausen, Field]**

WHAT IS NEURALLY PLAUSIBLE, ANYWAYS?

Our update rule (essentially) inherits a neural implementation from **[Olshausen, Field]**

However there are many competing theories for what constitutes a **plausible** neural implementation

e.g. nonnegative outputs, no bidirectional links, etc…

WHAT IS NEURALLY PLAUSIBLE, ANYWAYS?

Our update rule (essentially) inherits a neural implementation from **[Olshausen, Field]**

However there are many competing theories for what constitutes a **plausible** neural implementation

e.g. nonnegative outputs, no bidirectional links, etc…

But ours is **online**, **local** and **Hebbian**, all of which are basic properties to require

The surprise is that such simple building blocks can find **globally optimal** solutions to **highly non-trivial** algorithmic problems!

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:

$$\min_{\hat{A},\ \text{coln-sparse}\ \hat{X}} \quad E(\hat{A}, \hat{X}) = \left\| B - \hat{A}\,\hat{X} \right\|_F^2$$

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:

$$\min_{\hat{A},\ \text{coln-sparse}\ \hat{X}} \quad E(\hat{A}, \hat{X}) = \left\| B - \hat{A}\,\hat{X} \right\|_F^2$$

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?

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

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?

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

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?

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

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)** approximates the gradient of this function

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^*, 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:

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

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

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

This follows immediately from the usual proof…

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

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

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

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

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

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

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

Decoding Lemma: If $\hat{A}$ is 1/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

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

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

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

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

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

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

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

$$A_j - \hat{A}_j$$

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

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

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

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

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

$$\hat{A}_j \longleftarrow \hat{A}_j + \xi\,(I - \hat{A}_j\hat{A}_j^T\,)A_j + \xi\,\underbrace{\mathbf{E}_R[\hat{A}_R\hat{A}_R^T]A_j}_{\text{systemic bias}} + \text{error}$$

$A_j - \hat{A}_j$

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

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

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

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

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

$$\hat{A}_j \leftarrow \hat{A}_j + \xi\,(I - \hat{A}_j\hat{A}_j^T)A_j + \underbrace{\xi\,\mathbf{E}_R[\hat{A}_R\hat{A}_R^T]A_j}_{\text{systemic bias}} + \text{error}$$

$A_j - \hat{A}_j$

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

Auxiliary Lemma: $\|\hat{A} - A\| \leq 2$, remains true throughout if η is small enough and q is large enough

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \hat{A}\hat{x})\mathrm{sgn}(\hat{x}_j)]$$

is the expected update to $\hat{A}_j$.

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \widehat{A}\widehat{x})\mathrm{sgn}(\widehat{x}_j)]$$

is the expected update to $\widehat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x.

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \widehat{A}\widehat{x})\mathrm{sgn}(\widehat{x}_j)]$$

is the expected update to $\widehat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x.

$$g_j = E[(b - \widehat{A}\widehat{x})\mathrm{sgn}(\widehat{x}_j)\ 1_F] + E[(b - \widehat{A}\widehat{x})\mathrm{sgn}(\widehat{x}_j)\ 1_{\overline{F}}]$$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \hat{A}\hat{x})\operatorname{sgn}(\hat{x}_j)]$$

is the expected update to $\hat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x.

$$g_j = E[(b - \hat{A}\hat{x})\operatorname{sgn}(\hat{x}_j)\ 1_F] \pm \zeta$$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \hat{A}\hat{x})\mathrm{sgn}(\hat{x}_j)]$$

is the expected update to $\hat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x.

$$g_j = E[(b - \hat{A}\hat{x})\mathrm{sgn}(x_j)\ 1_F] \pm \zeta$$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \hat{A}\hat{x})\mathrm{sgn}(\hat{x}_j)]$$

is the expected update to $\hat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x.

$$g_j = E[(b - \hat{A}\,\mathrm{threshold}(\hat{A}^T b))\ \mathrm{sgn}(x_j)\ 1_F] \pm \zeta$$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \widehat{A}\widehat{x})\text{sgn}(\widehat{x}_j)]$$

is the expected update to $\widehat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x. Let S = supp(x).

$$g_j = E[(b - \widehat{A}_S\widehat{A}_S^T b)\ \text{sgn}(x_j)\ 1_F] \pm \zeta$$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \widehat{A}\widehat{x})\mathrm{sgn}(\widehat{x}_j)]$$

is the expected update to $\widehat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x. Let S = supp(x).

$$\begin{aligned} g_j = {} & E[(b - \widehat{A}_S\widehat{A}_S^T b)\ \mathrm{sgn}(x_j)] \\ & - E[(b - \widehat{A}_S\widehat{A}_S^T b)\ \mathrm{sgn}(x_j)\ 1_{\overline{F}}] \pm \zeta \end{aligned}$$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \hat{A}\hat{x})\mathrm{sgn}(\hat{x}_j)]$$

is the expected update to $\hat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x. Let S = supp(x).

$$g_j = E[(I - \hat{A}_S\hat{A}_S^T)Ax\ \mathrm{sgn}(x_j)] \pm \zeta$$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \widehat{A}\widehat{x})\text{sgn}(\widehat{x}_j)]$$

is the expected update to $\widehat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x. Let S = supp(x).

$$g_j = E[(I - \widehat{A}_S\widehat{A}_S^T)Ax\ \text{sgn}(x_j)] \pm \zeta$$

$$= E_S E_{x_S}[[(I - \widehat{A}_S\widehat{A}_S^T)Ax\ \text{sgn}(x_j)]|S] \pm \zeta$$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \hat{A}\hat{x})\mathrm{sgn}(\hat{x}_j)]$$

is the expected update to $\hat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x. Let S = supp(x).

$$g_j = E[(I - \hat{A}_S\hat{A}_S^T)Ax\ \mathrm{sgn}(x_j)] \pm \zeta$$

$$= E_S E_{x_S}[[(I - \hat{A}_S\hat{A}_S^T)Ax\ \mathrm{sgn}(x_j)]\,|\,S] \pm \zeta$$

$$= p_j\ E_S[(I - \hat{A}_S\hat{A}_S^T)A_j] \pm \zeta$$

where $p_j = E[x_j\ \mathrm{sgn}(x_j)\,|\,j \text{ in } S]$.

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \widehat{A}\widehat{x})\mathrm{sgn}(\widehat{x}_j)]$$

is the expected update to $\widehat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x. Let S = supp(x).

$$g_j = E[(I - \widehat{A}_S\widehat{A}_S^T)Ax\ \mathrm{sgn}(x_j)] \pm \zeta$$

$$= E_S E_{x_S}[[(I - \widehat{A}_S\widehat{A}_S^T)Ax\ \mathrm{sgn}(x_j)]\,|\,S] \pm \zeta$$

$$= p_j\ E_S[(I - \widehat{A}_S\widehat{A}_S^T)A_j] \pm \zeta$$

where $p_j = E[x_j\ \mathrm{sgn}(x_j)\,|\,j \text{ in } S]$. Let $q_j = \Pr[j \text{ in } S]$, $q_{i,j} = \Pr[i,j \text{ in } S]$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \hat{A}\hat{x})\mathrm{sgn}(\hat{x}_j)]$$

is the expected update to $\hat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x. Let S = supp(x).

$$g_j = E[(I - \hat{A}_S\hat{A}_S^T)Ax\ \mathrm{sgn}(x_j)] \pm \zeta$$

$$= E_S E_{x_S}[[(I - \hat{A}_S\hat{A}_S^T)Ax\ \mathrm{sgn}(x_j)]|S] \pm \zeta$$

$$= p_j\ E_S[(I - \hat{A}_S\hat{A}_S^T)A_j] \pm \zeta$$

where $p_j = E[x_j\ \mathrm{sgn}(x_j)|j \text{ in } S]$. Let $q_j = \Pr[j \text{ in } S]$, $q_{i,j} = \Pr[i,j \text{ in } S]$ then

$$= p_j q_j\ (I - \hat{A}_j\hat{A}_j^T)A_j + p_j\ \hat{A}_{-j}\mathrm{diag}(q_{i,j})\hat{A}_{-j}^T\ A_j \pm \zeta$$

Proof: Let ζ denote any vector whose norm is negligible (say, $n^{-\omega(1)}$).

$$g_j = E[(b - \hat{A}\hat{x})\mathrm{sgn}(\hat{x}_j)]$$

is the expected update to $\hat{A}_j$. Let 1_F be the indicator of the event that decoding recovers the signs of x. Let S = supp(x).

$$g_j = E[(I - \hat{A}_S\hat{A}_S^T)Ax\ \mathrm{sgn}(x_j)] \pm \zeta$$

$$= E_S E_{x_S}[[(I - \hat{A}_S\hat{A}_S^T)Ax\ \mathrm{sgn}(x_j)]|S] \pm \zeta$$

$$= p_j\ E_S[(I - \hat{A}_S\hat{A}_S^T)A_j] \pm \zeta$$

where $p_j = E[x_j\ \mathrm{sgn}(x_j)|j \text{ in } S]$. Let $q_j = \Pr[j \text{ in } S]$, $q_{i,j} = \Pr[i,j \text{ in } S]$ then

$$= p_j q_j\ (I - \hat{A}_j\hat{A}_j^T)A_j + p_j\ \hat{A}_{-j}\mathrm{diag}(q_{i,j})\hat{A}_{-j}^T\ A_j \pm \zeta$$ ■

AN INITIALIZATION PROCEDURE

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

AN INITIALIZATION PROCEDURE

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

Repeat: **(1)** Choose samples b, b′

AN INITIALIZATION PROCEDURE

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

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 δ-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 $\widehat{A}$ that is column-wise δ-close to A for $\delta \leq 1/\text{polylog}(n)$, $\|\widehat{A} - A\| \leq 2$

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

Key Lemma: If Ax = b and Ax' = b', then condition **(3)** is satisfied if and only if $\text{supp}(x) \cap \text{supp}(x') = \{j\}$

AN INITIALIZATION PROCEDURE

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

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

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 δ-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 $\delta \leq 1/\mathrm{polylog}(n)$, can be neurally implemented with **Oja's Rule**

Earlier analyses of alternating minimization for $\delta \leq 1/\mathrm{poly}(n)$ 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 δ ≤ 1/polylog(n), can be neurally implemented with **Oja's Rule**

Earlier analyses of alternating minimization for δ ≤ 1/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 δ ≤ 1/polylog(n), can be neurally implemented with **Oja's Rule**

Earlier analyses of alternating minimization for δ ≤ 1/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)

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 **systematically** design/analyze new update rules

FURTHER RESULTS

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

We can use our framework to **systematically** design/analyze 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 **systematically** design/analyze new update rules

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

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

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

FURTHER RESULTS

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

We can use our framework to **systematically** design/analyze new update rules

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

(1) $\widehat{x}_j^{(i)} = \text{threshold}(\widehat{C}_j^T b^{(i)})$

where $\widehat{C}_j = [\text{Proj}_{\widehat{A}_j^\perp}(\widehat{A}_1), \text{Proj}_{\widehat{A}_j^\perp}(\widehat{A}_2), \ldots \widehat{A}_j \ldots \text{Proj}_{\widehat{A}_j^\perp}(\widehat{A}_m)]$

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

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?