10707
Deep Learning: Spring 2020

Andrej Risteski

Machine Learning Department

Lecture 14:
Simplest of representation learners: autoencoders and sparse coding

Unsupervised learning

Learning from data **without** labels.

What can we hope to do:

Task A: Fit a parametrized **structure** (e.g. clustering, low-dimensional subspace, manifold) to data to reveal something meaningful about data. (**Structure learning**)

Task B: Learn a (parametrized) **distribution** *close* to data generating distribution. (**Distribution learning**)

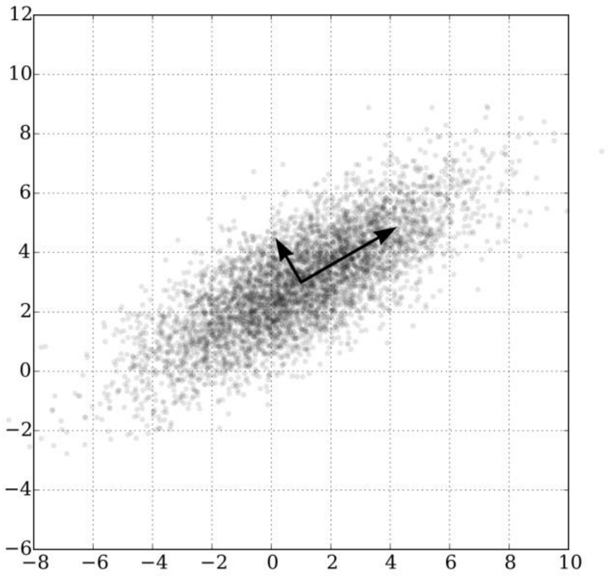
Task C: Learn a (parametrized) distribution that implicitly reveals an **"embedding"/"representation"** of data for downstream tasks. (**Representation/feature learning**)

Entangled! The "structure" and "distribution" often reveals an embedding.

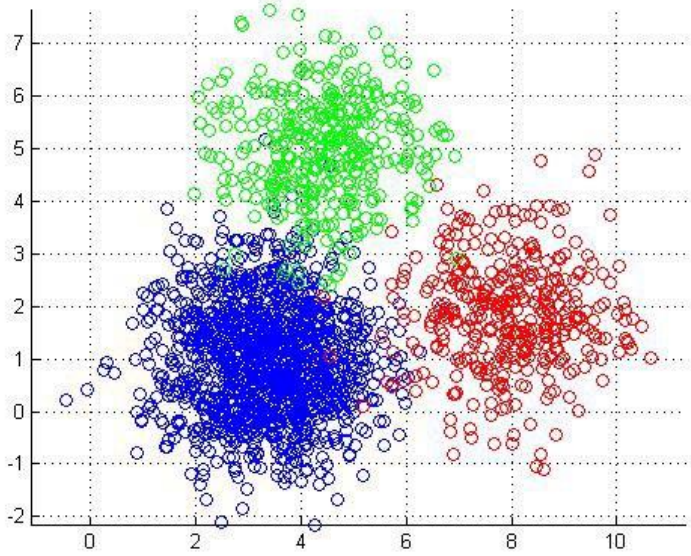
Structure learning

Fit a parametrized **structure** (e.g. clustering, low-dimensional subspace) to data to reveal something meaningful about data.

PCA(principal component analysis), direction of highest variance



Clustering



The simplest of representation learners

Sparse coding: learn features, s.t. each input can be written as a *sparse linear combination* of some of these features.

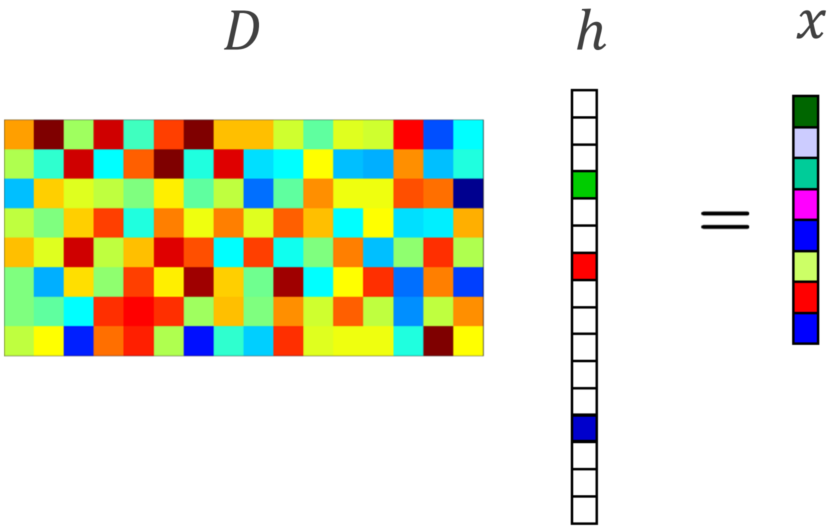
Originally made famous by *Olshausen and Field*, ‘96 as a model for how early visual processing works (edge detection etc.)

Autoencoders: learn encoding with some constraints (e.g. functional form, sparsity, denoising ability) from which the inputs can be approximately reconstructed.

Sparse coding

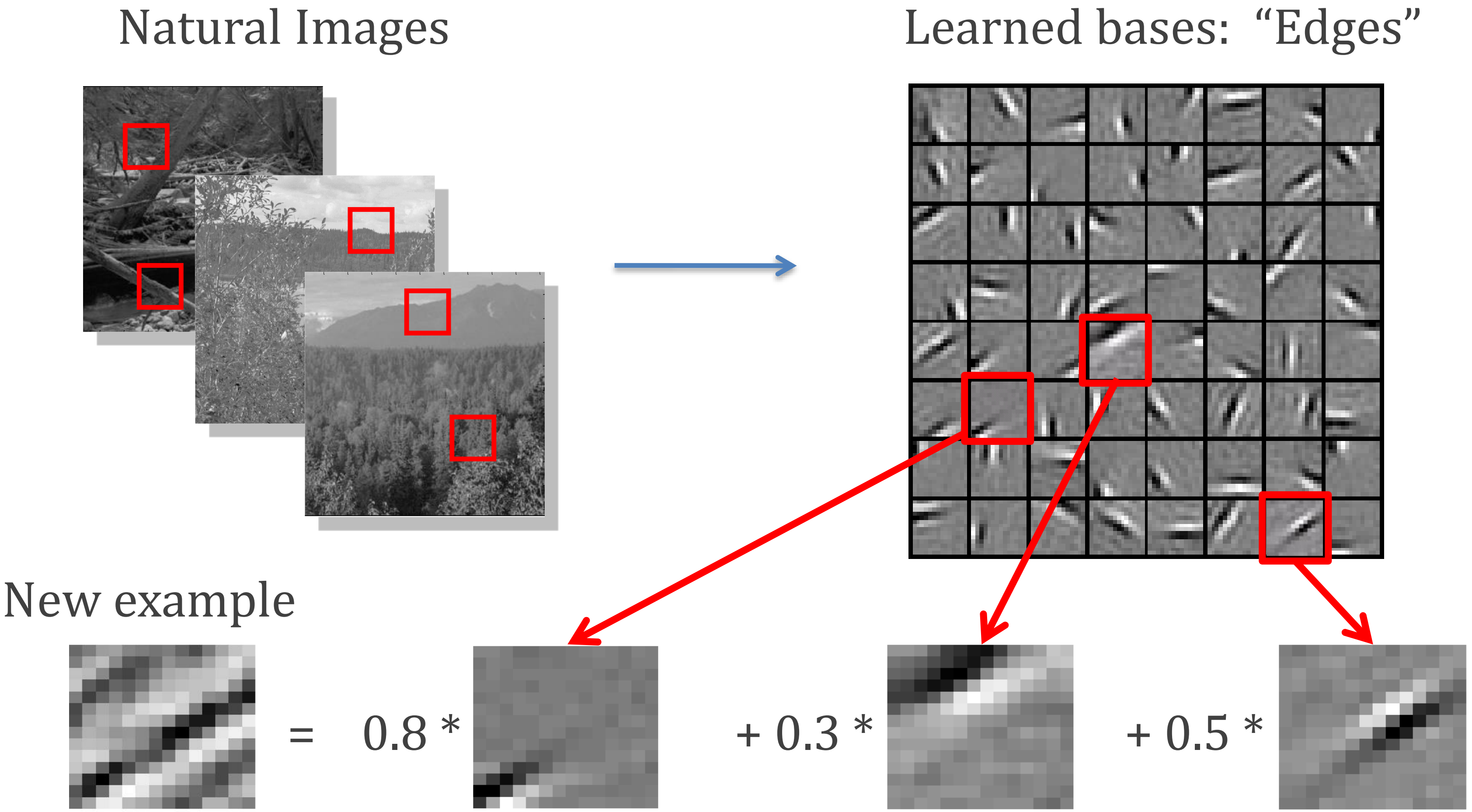
Goal: learn a *dictionary* D of features, s.t. each sample x is (approximately) writeable as a *sparse* (i.e. mostly zeros) linear combination of these features.

$$\forall x: \quad x \approx Dh, \qquad ||h||_0 \text{ small}$$



h is the representation of sample x

Sparse coding



Natural Images

Learned bases: “Edges”

New example

= 0.8 * + 0.3 * + 0.5 *

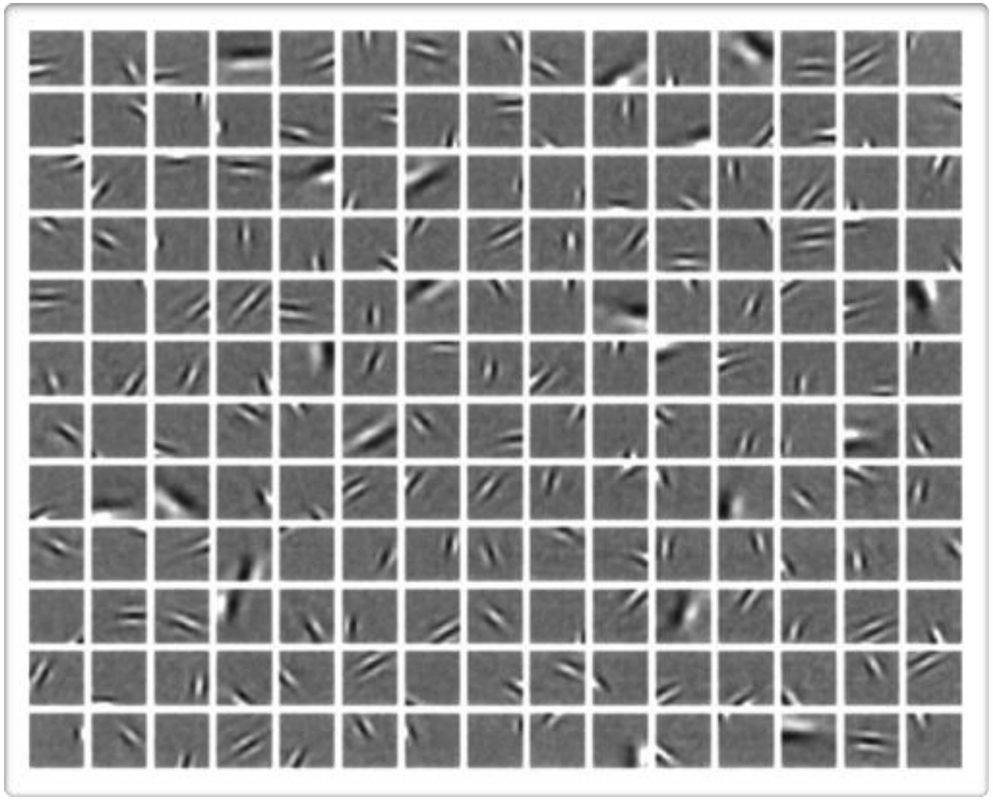
[0, 0, … **0.8**, …, **0.3**, …, **0.5**, …] = coefficients (feature representation)

Slide Credit: Honglak Lee

Relationship to V1

When trained on natural image patches

- the dictionary columns ("atoms") look like edge detectors
- each atom is "tuned" to a particular position, orientation and spatial frequency
- V1 neurons in the mammalian brain have a similar behavior



Emergence of simple-cell receptive field properties by learning a sparse code of natural images. Olshausen and Field, 1996.

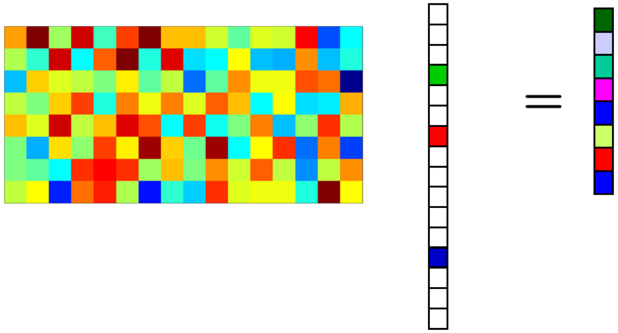
Sparse coding

Historical motivation: in signal processing, it's common to have a *fixed* dictionary D (typically, these are Fourier-basis inspired features), that's hand-crafted for the domain.

Why is this useful?: think of x as an image. It takes a lot of bits of information to write down x in the standard basis (exponential in size)

Wasteful: most vectors of numbers of image dimensions are not "real images". There ought to be better bases… (Fourier, wavelet, …)

In the right basis, image ought to be writeable as a combination of a *small (i.e. sparse) combination* of elements. Need much less bits to represent image (~k log d, since there are d^k possible supports*)*



Sparse coding

Historical motivation: in signal processing, it's common to have a *fixed* dictionary D (typically, these are Fourier-basis inspired features), that's hand-crafted for the domain.

Why is this useful?: think of x as an image. It takes a lot of bits of information to write down x in the standard basis (exponential in size)

Wasteful: most vectors of numbers of image dimensions are not "real images". There ought to be better bases… (Fourier, wavelet, …)

In the right basis, image ought to be writeable as a combination of a *small (i.e. sparse) combination* of elements. Need much less bits to represent image (~k log d, since there are d^k possible supports*)*

Sparse coding is compressive sensing, where we are learning the dictionary as well. (Fits spirit of deep learning!)

Algorithms

How do we fit D?

Obvious first try:

Reconstruction: $\widehat{\mathbf{x}}^{(t)}$

Sparsity vs.
reconstruction control

$$\min_{\mathbf{D}} \frac{1}{T}\sum_{t=1}^{T} \underbrace{\min_{\mathbf{h}^{(t)}} \frac{1}{2}||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2}_{\text{Reconstruction error}} + \underbrace{\lambda||\mathbf{h}^{(t)}||_{\,0}}_{\text{Sparsity penalty}}$$

Can’t quite take gradients: l_0 is either flat (gradients are 0) or not differentiable.

Algorithms

How do we fit D?

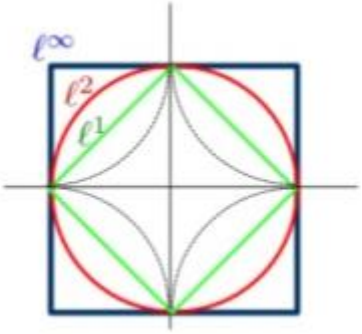
Typical relaxation:

Reconstruction: $\widehat{\mathbf{x}}^{(t)}$

Sparsity vs. reconstruction control

$$\min_{\mathbf{D}} \frac{1}{T} \sum_{t=1}^{T} \min_{\mathbf{h}^{(t)}} \underbrace{\frac{1}{2}||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2}_{\text{Reconstruction error}} + \underbrace{\lambda||\mathbf{h}^{(t)}||_1}_{\text{Sparsity penalty}}$$

l_1 is the convex envelope of l_0 : the closest function that is convex.



Algorithms

How do we fit D?

Typical relaxation:

Reconstruction: $\widehat{\mathbf{x}}^{(t)}$

Sparsity vs. reconstruction control

$$\min_{\mathbf{D}} \frac{1}{T} \sum_{t=1}^{T} \underbrace{\min_{\mathbf{h}^{(t)}} \frac{1}{2} ||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2}_{\text{Reconstruction error}} + \underbrace{\lambda ||\mathbf{h}^{(t)}||_1}_{\text{Sparsity penalty}}$$

- We also constrain the columns of D to be of norm 1
- Otherwise, we can scale D up, scale h's down, which improves sparsity penalty, but clearly doesn't encourage sparsity.

Inference

Given dictionary D , how do we compute $\mathbf{h}(\mathbf{x}^{(t)})$?

- We need to optimize:

$$l(\mathbf{x}^{(t)}) = \frac{1}{2}||\mathbf{x}^{(t)} - \mathbf{D}\;\mathbf{h}^{(t)}||_2^2 + \lambda||\mathbf{h}^{(t)}||_1$$

- *Usual candidate*: gradient descent

$$\nabla_{\mathbf{h}^{(t)}} l(\mathbf{x}^{(t)}) = \mathbf{D}^\top(\mathbf{D}\;\mathbf{h}^{(t)} - \mathbf{x}^{(t)}) + \lambda\,\mathrm{sign}(\mathbf{h}^{(t)})$$

- *Issue*: l_1 norm not differentiable at 0: very unlikely for gradient descent to "land" on $h_k^{(t)} = 0$ (even if it's the solution)
- Solution: if $h_k^{(t)}$ changes sign, clamp to 0.
- Sometimes called **ISTA (Iterative Shrinkage Thresholding Algorithm)**

Slide Credit: Russ Salakhutdinov

Inference

Given dictionary D , how do we compute $\mathbf{h}(\mathbf{x}^{(t)})$?

- We need to optimize:

$$l(\mathbf{x}^{(t)}) = \frac{1}{2}||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2 + \lambda||\mathbf{h}^{(t)}||_1$$

Each hidden unit update would be performed as follows:

- $h_k^{(t)} \Longleftarrow h_k^{(t)} - \alpha(\mathbf{D}_{\cdot,k})^\top(\mathbf{D}\;\mathbf{h}^{(t)} - \mathbf{x}^{(t)})$
- If $\mathrm{sign}(h_k^{(t)}) \neq \mathrm{sign}(h_k^{(t)} - \alpha\,\lambda\,\mathrm{sign}(h_k^{(t)}))$ then $h_k^{(t)} \Longleftarrow 0$
- Else $h_k^{(t)} \Longleftarrow h_k^{(t)} - \alpha\,\lambda\,\mathrm{sign}(h_k^{(t)})$

Update from reconstruction

Update sparsity term

Slide Credit: Russ Salakhutdinov

Inference: simple examples

Let’s assume that $x = Dh^*$, for an orthogonal D ($D^T D = I$)

Let’s assume the non-zero entries of h are bounded away from 0, namely $|h_i| \geq \tau$ if $h_i \neq 0$.

The ISTA update looks like:

$$h \leftarrow h - \alpha\left(D^T(Dh - x)\right)$$

$$\text{If } \operatorname{sgn}(h_k) \neq \operatorname{sgn}(h_k - \alpha\lambda \operatorname{sgn}(h_k)) \Rightarrow h_k \leftarrow 0$$

Set $\alpha = 1, \lambda < \tau$. Then, we have:

$$h \leftarrow D^T x = D^T D h^* = h^*$$

$$\forall k, \operatorname{sgn}(h_k) = \operatorname{sgn}(h_k - \lambda \operatorname{sgn}(h_k))$$

Done in one step!!

Inference: simple examples

Let's assume that $x = Dh^* + \epsilon$, for an orthogonal D ($D^T D = I$), s.t. $||\epsilon||_2 \leq \frac{\tau}{4}$. Let's assume the non-zero entries of h satisfy $|h_i| \geq \tau$ if $h_i \neq 0$.

The ISTA update looks like:

$$h \leftarrow h - \alpha\big(D^T(Dh - x)\big)$$

$$\text{If } \operatorname{sgn}(h_k) \neq \operatorname{sgn}(h_k - \alpha\lambda \operatorname{sgn}(h_k)) \Rightarrow h_k \leftarrow 0$$

Set $\alpha = 1, \lambda = \tau/2$. Then, we have: $h \leftarrow D^T x = D^T(Dh^* + \epsilon) = h^* + D^T\epsilon$

Note that $\langle D_{.,k}, \epsilon\rangle \leq ||\epsilon||_2$ by orthogonality, so $h_i = h_i^* + \delta_i, |\delta_i| \leq \frac{\tau}{4}$

$$\left.\begin{array}{l} \text{If } h_i^* \neq 0, |\mathrm{h_i}| \geq \dfrac{3\tau}{4} \Rightarrow \operatorname{sgn}(h_i) = \operatorname{sgn}(h_i - \tau/2 \operatorname{sgn}(h_i)) \\ \text{If } h_i^* \neq 0, |\mathrm{h_i}| \leq \dfrac{\tau}{4} \Rightarrow \operatorname{sgn}(h_i) \neq \operatorname{sgn}(h_i - \tau/2 \operatorname{sgn}(h_i)) \end{array}\right\} \textit{Done in one step!!}$$

Inference: simple examples

Let's assume that $x = Dh^* + \epsilon$, for a $D \in \mathbb{R}^{d\times D}, D \gg d$ and $(D^TD)_{ii} = 1, (D^TD)_{ij} \leq \mu$ (i.e. the columns of D are close to orthogonal). Furthermore, $||\epsilon||_2 \leq \frac{\tau}{8}$ and the non-zero entries of h satisfy M $\geq |h_i| \geq \tau$ if $h_i \neq 0$ and μ, M are s.t. $||h^*||_0 \mu M \leq \frac{\tau}{8}$

The ISTA update looks like: $h \leftarrow h \; - \alpha\big(D^T(Dh \; - x)\big)$

$$\text{If } \operatorname{sgn}(h_k) \neq \operatorname{sgn}(h_k \; - \alpha\lambda \operatorname{sgn}(h_k)) \Rightarrow h_k \leftarrow 0$$

Set $\alpha = 1, \lambda = \tau/2$. Then, we have: $h \leftarrow D^Tx = D^T(Dh^* + \epsilon) = D^TD\, h^* + D^T\epsilon$

Consider first term: $(D^TD\, h^*)_k = \sum_j (D^TD)_{kj} h_j^* \; = h_k^* + \sum_{j:h_j^* \neq 0} (D^TD)_{kj} h_j^*$

The last part has: $|\sum_{j:h_j^*\neq 0}(D^TD)_{kj}h_j^*| \; \leq ||h^*||_0 \mu M \leq \frac{\tau}{8}$. As before, $\langle D_{.,k}, \epsilon\rangle \leq ||\epsilon||_2 \leq \frac{\tau}{8}$.

We finish as before, $h_i \; = h_i^* + \delta_i, |\delta_i| \leq \frac{\tau}{4}$, so:

$$\left.\begin{array}{l} \text{If } h_i^* \neq 0, |\mathrm{h_i}| \leq \dfrac{\tau}{4} \Rightarrow \; \operatorname{sgn}(h_i) \neq \operatorname{sgn}(h_i \; - \tau/2 \operatorname{sgn}(h_i)) \\[2ex] \text{If } h_i^* \neq 0, |\mathrm{h_i}| \geq \dfrac{3\tau}{4} \Rightarrow \; \operatorname{sgn}(h_i) = \operatorname{sgn}(h_i \; - \tau/2 \operatorname{sgn}(h_i)) \end{array}\right\} \textit{Done in one step!!}$$

Dictionary learning algorithm

Given that we have a mechanism for finding good h's for a fixed dictionary, we can do the same thing we did in the EM algorithm: alternate optimizing.

Keeping the h's fixed, perform gradient descent for D:

- Perform gradient update of D

$$\mathbf{D} \Longleftarrow \mathbf{D} + \alpha \frac{1}{T} \sum_{t=1}^{T} (\mathbf{x}^{(t)} - \mathbf{D}\, \mathbf{h}(\mathbf{x}^{(t)}))\, \mathbf{h}(\mathbf{x}^{(t)})^{\top}$$

- Renormalize the columns of D
- For each column of D:

$$\mathbf{D}_{\cdot,j} \Longleftarrow \frac{\mathbf{D}_{\cdot,j}}{||\mathbf{D}_{\cdot,j}||_2}$$

Slide Credit: Russ Salakhutdinov

Dictionary learning algorithm

Given that we have a mechanism for finding good h's for a fixed dictionary, we can do the same thing we did in the EM algorithm: alternate optimizing.

```
While D has not converged:
    ➢ find the sparse codes $\mathbf{h}(\mathbf{x}^{(t)})$ for all $\mathbf{x}^{(t)}$ in the training set with ISTA
    ➢ Update the dictionary by running gradient descent for D.
```

How is this analyzed?

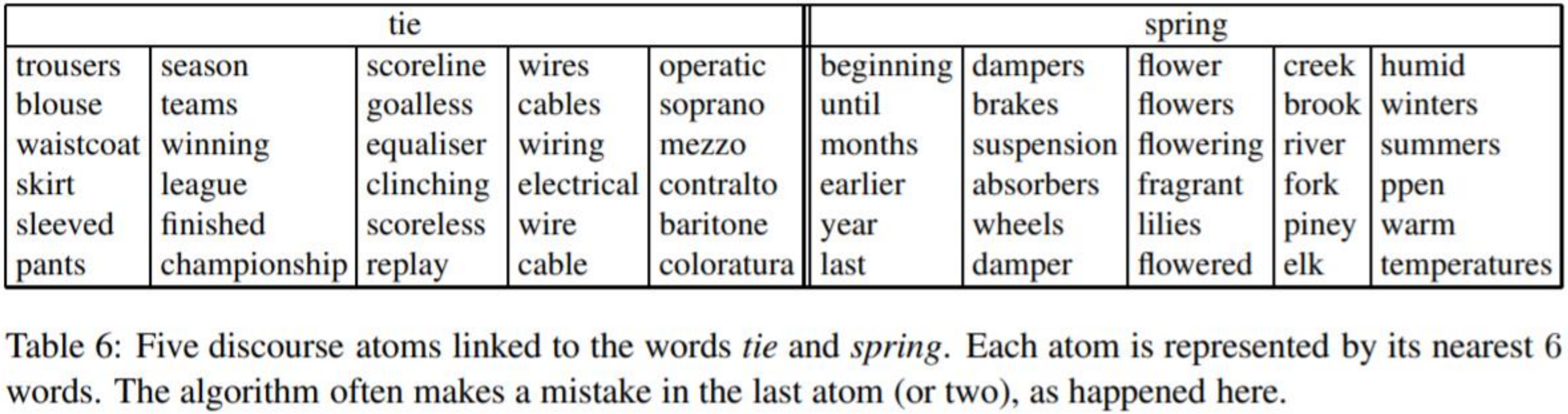
In the beginning, the dictionary is way off – so our inference analyses don’t quite work.

Analyzing the dynamics of the algorithm is quite difficult: current results assume *warm starts:* the dictionary we initialize with is not too far from ground truth.

(Agarwal, Anandkumar, Jain, Netrapalli ‘14, Arora, Ge, Ma, Moitra ‘15, Li, Liang, Risteski ‘16, Chatterji, Bartlett ‘17)

Analyzing dynamics from random start is still wide open.

Some applications



tie					spring				
trousers	season	scoreline	wires	operatic	beginning	dampers	flower	creek	humid
blouse	teams	goalless	cables	soprano	until	brakes	flowers	brook	winters
waistcoat	winning	equaliser	wiring	mezzo	months	suspension	flowering	river	summers
skirt	league	clinching	electrical	contralto	earlier	absorbers	fragrant	fork	ppen
sleeved	finished	scoreless	wire	baritone	year	wheels	lilies	piney	warm
pants	championship	replay	cable	coloratura	last	damper	flowered	elk	temperatures

Table 6: Five discourse atoms linked to the words *tie* and *spring*. Each atom is represented by its nearest 6 words. The algorithm often makes a mistake in the last atom (or two), as happened here.

Finding the different meanings of polysemous words
(Arora, Li, Liang, Ma, Risteski '18)

Some applications

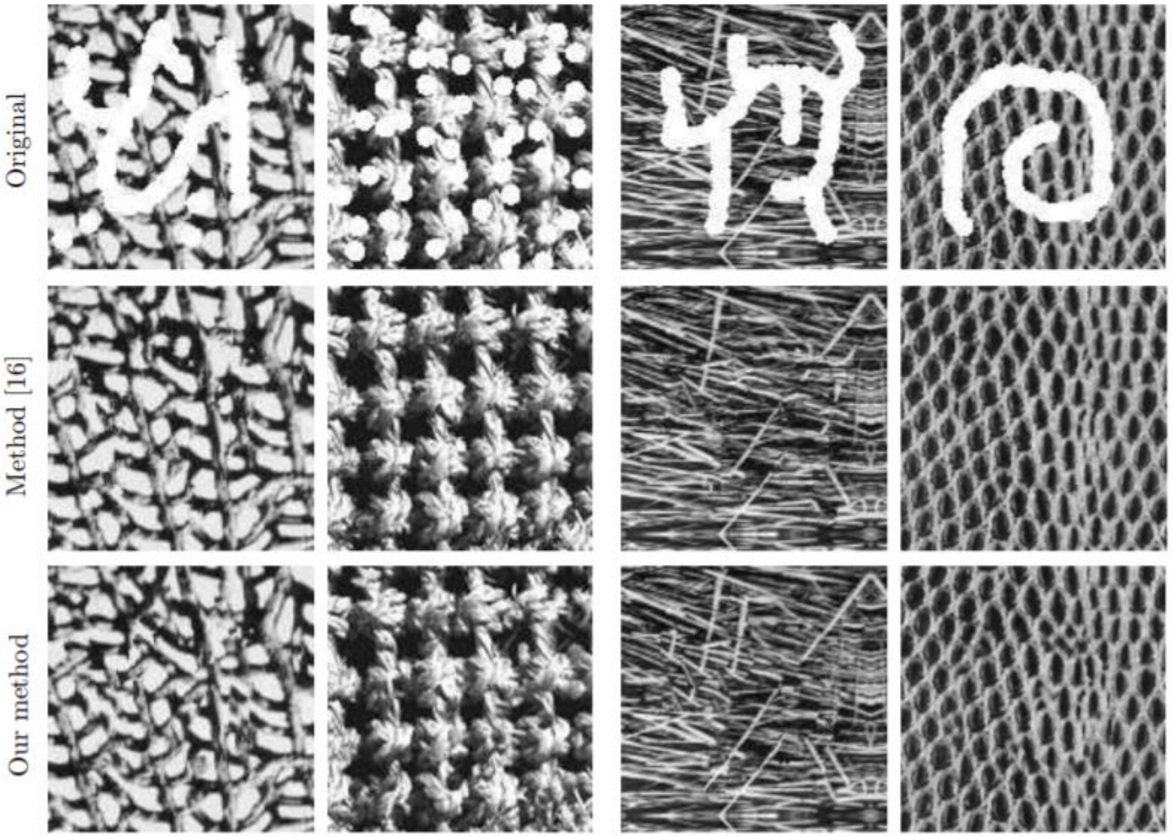
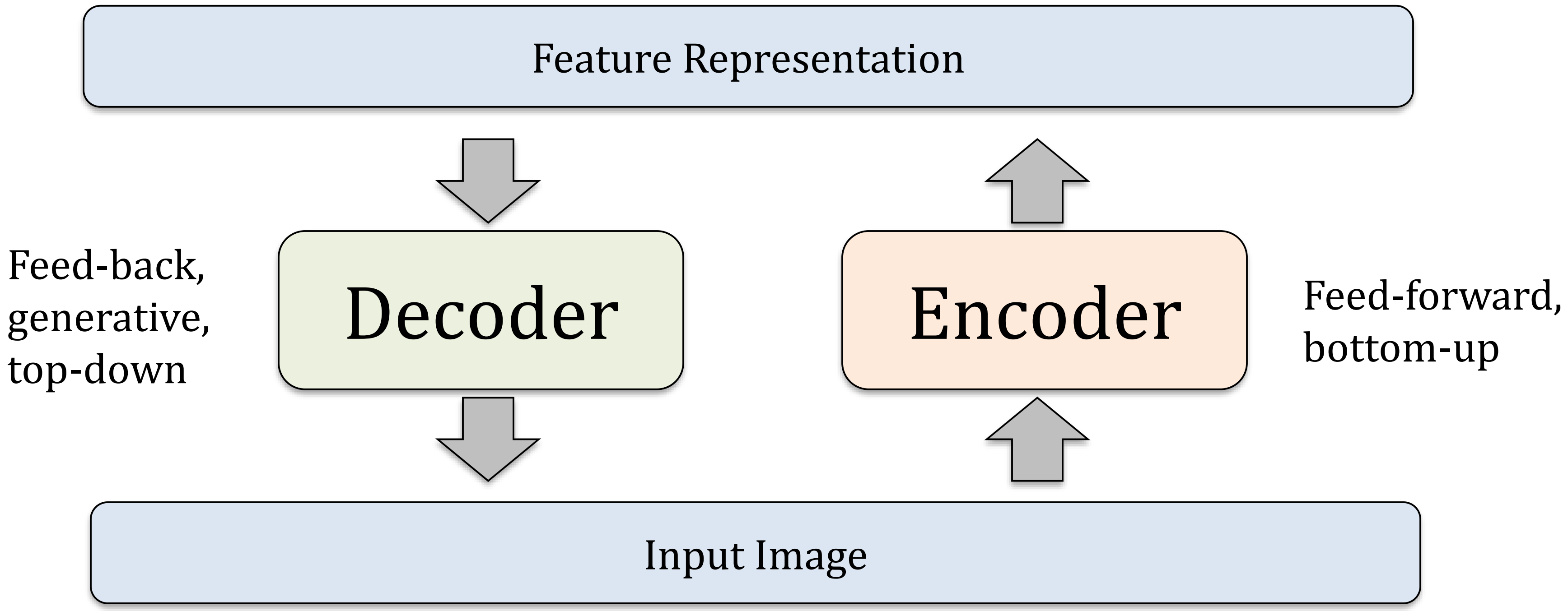


Figure 8: *Examples of inpainting using [16] and using the proposed method.*

Sparse Modeling of Textures
(Gabriel Peyré, '09)

Autoencoders

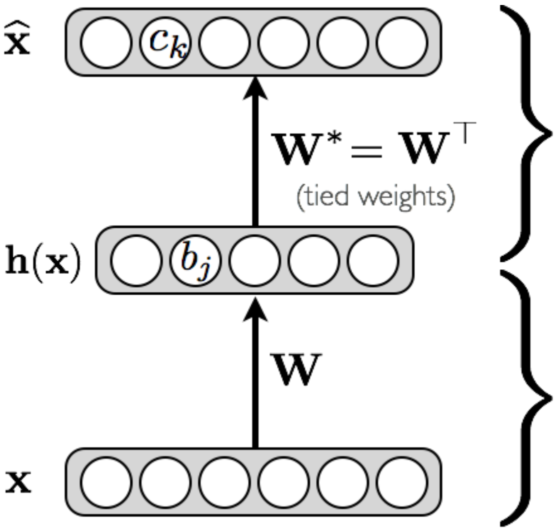
The idea behind autoencoders: learn features, s.t. input is reconstructable from them



- Details of what goes insider the encoder and decoder matter!
- Need *constraints* to **avoid learning an identity**.

Autoencoders

Some way to prevent identity:



- *Weight tying* of encoder/decoder. (Often magical!)
- *Smaller dimension* for latent variables
- Enforce *sparsity* of the latent representation
- Encourage decoder to be robust to adding noise to x. (*Denoising autoencoder*)
- Encode to distribution rather than pointmass. (*Variational autoencoder*)

Typical losses

Loss function for inputs between 0 and 1

$$l(f(\mathbf{x})) = -\sum_k \left(x_k \log(\widehat{x}_k) + (1 - x_k)\log(1 - \widehat{x}_k)\right)$$

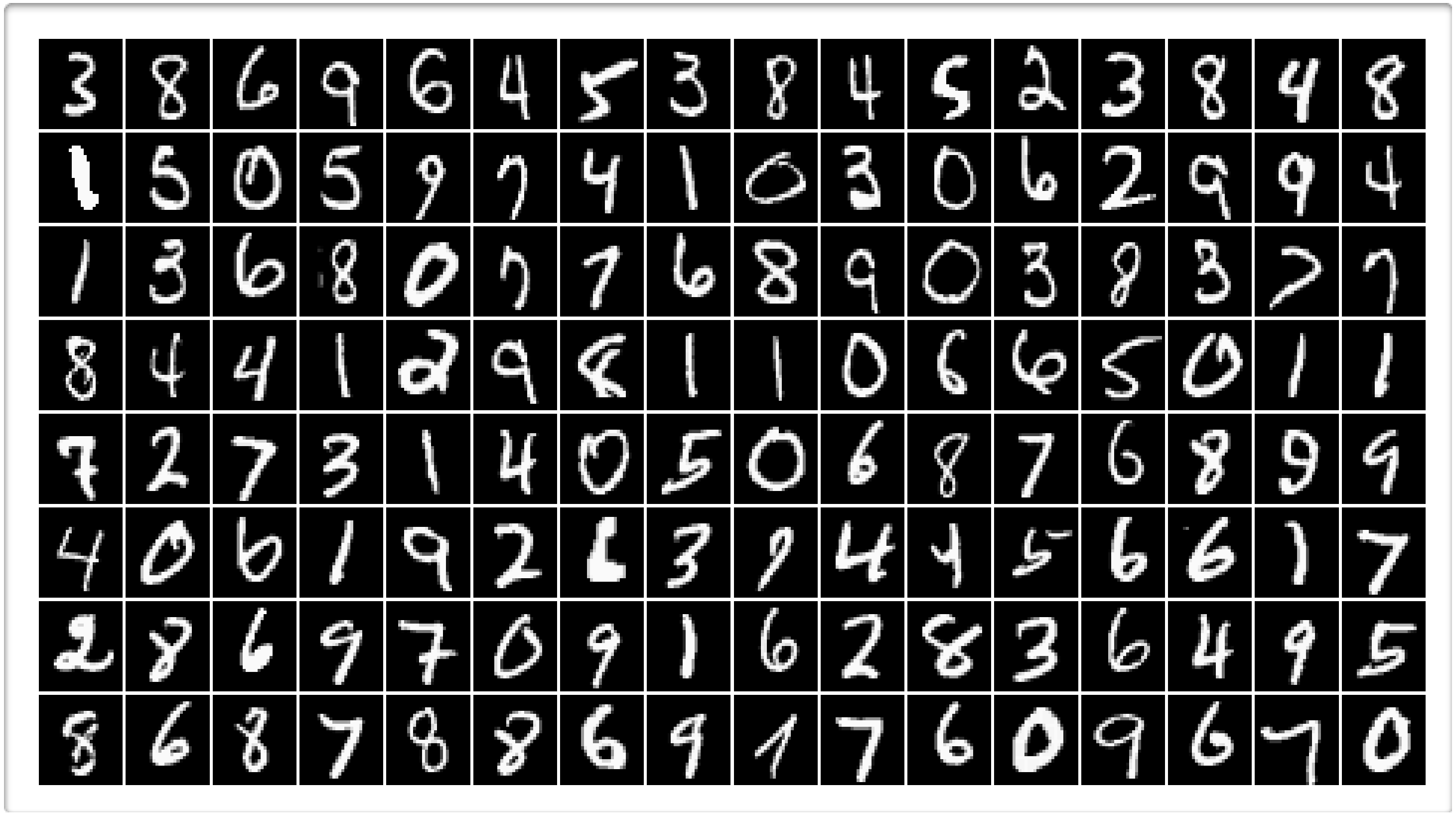
- *Cross-entropy error* $(f(\mathbf{x}) \equiv \widehat{\mathbf{x}})$

Loss function for real-valued inputs

$$l(f(\mathbf{x})) = \frac{1}{2}\sum_k (\widehat{x}_k - x_k)^2$$

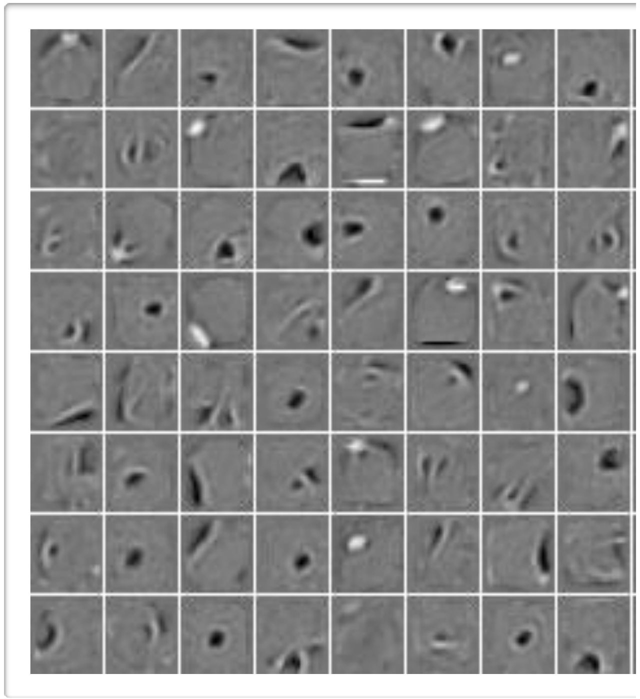
- l_2 error
- we use a linear activation function at the output

Example: Reconstructions on MNIST

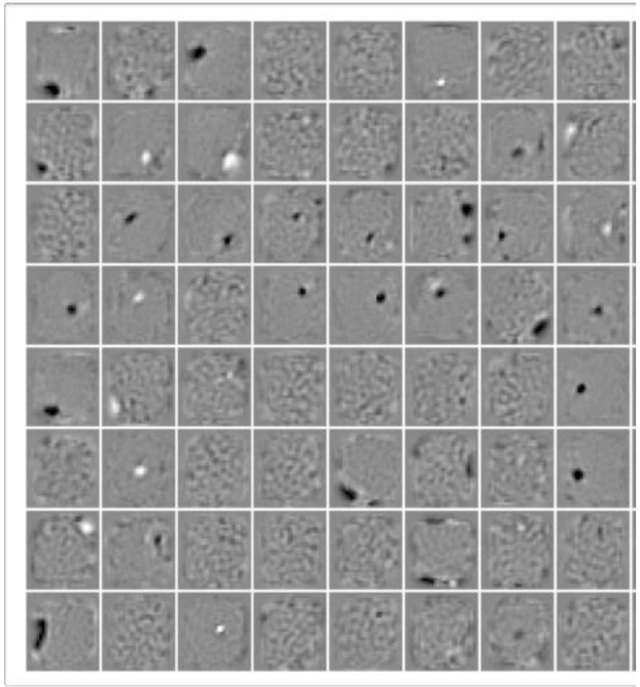


Learned Features

MNIST dataset:



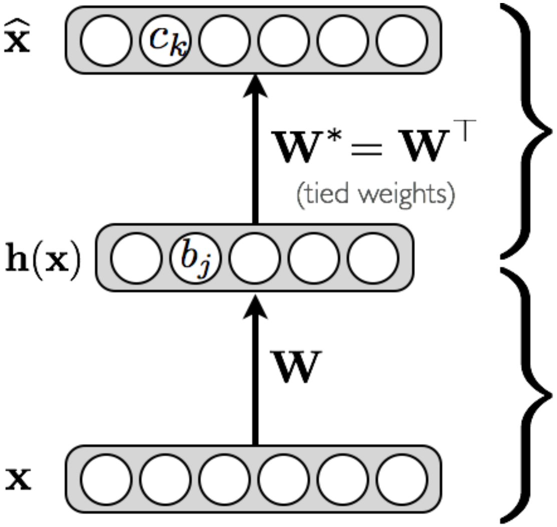
RBM



Autoencoder

(Larochelle et al., JMLR 2009)

Intuitions for weight tieing

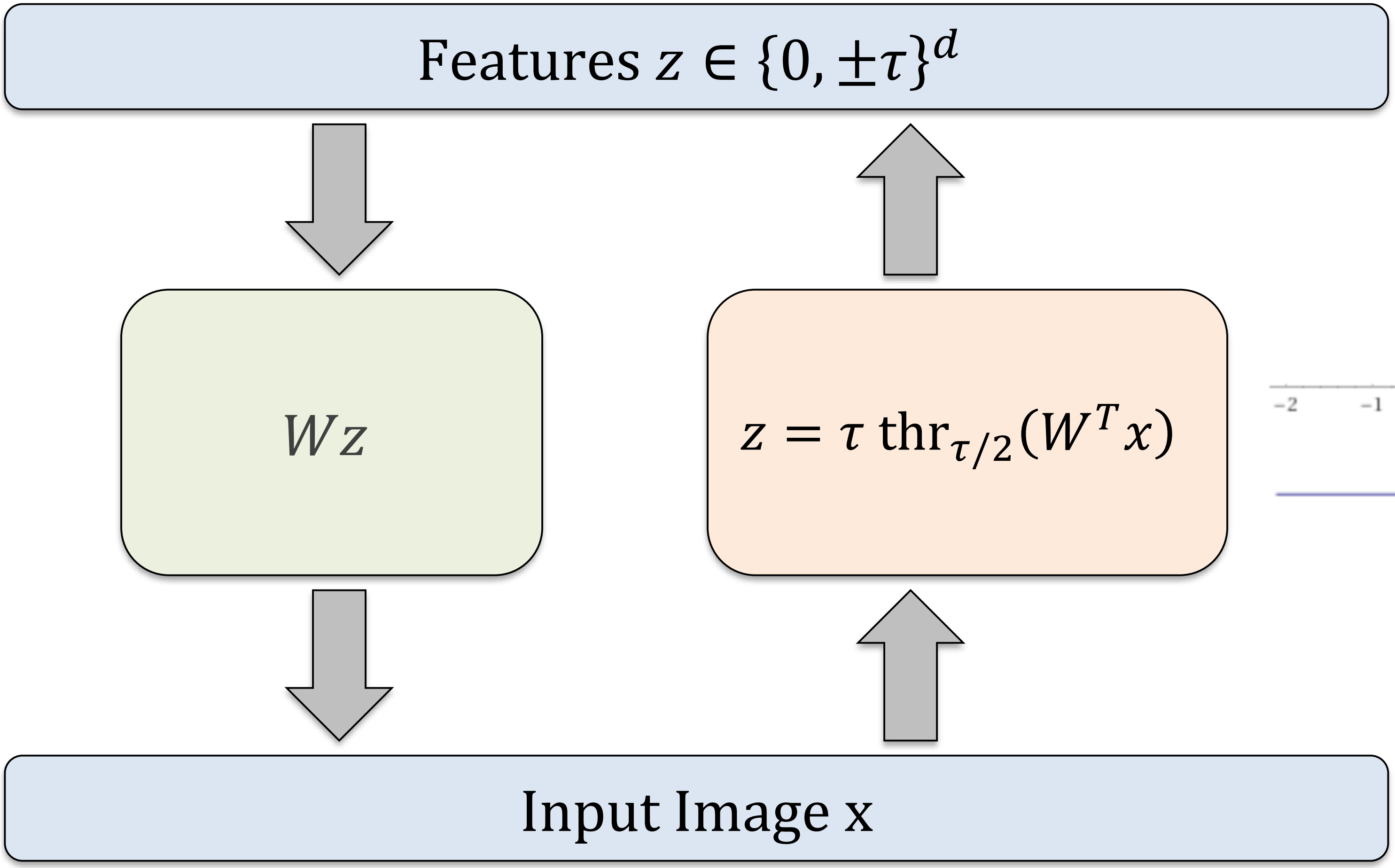


Original intuition: similar as doing 2 steps in a Gibbs sampler in RBM's.
(Though not randomized.)

Sharper intuition: one step of ISTA algorithm for dictionary learning!

Intuitions for weight tieing

Setup: sgn activations with weight tieing

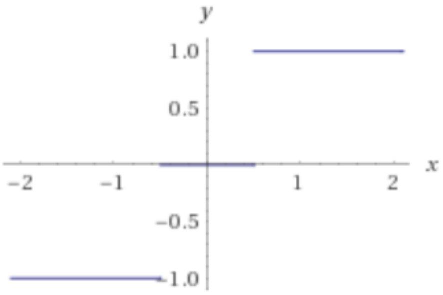


Features $z \in \{0, \pm\tau\}^d$

Wz

$z = \tau \, \mathrm{thr}_{\tau/2}(W^T x)$

Input Image x



Intuitions for weight tieing

Claim: if true x's satisfy x $= \ Wz + \epsilon$, for W orthogonal, $||\epsilon||_2 \leq \frac{\tau}{4}$, the above combination of encoder/decoders give a reconstruction error of at most $||\epsilon||_2$

Same calculation as doing one ISTA step!

Encoder produces: $\mathrm{thr}_{\tau/2}\ (W^T x) = \mathrm{thr}_{\tau/2}(W^T(Wz + \epsilon))$

$$= \mathrm{thr}_{\tau/2}(z + W^T\epsilon)$$

As $\langle W_{.,k}, \epsilon \rangle \leq ||\epsilon||_2$, encoder produces $z_i + \delta_i, |\delta_i| \leq \frac{\tau}{4}$.

If $|z_i| = \tau$, input to thr is $z_i + \delta_i \in \tau \pm \frac{\tau}{4}$, hence encoder produces z_i

If $z_i = 0$, input to thr is $z_i + \delta_i \in \pm\frac{\tau}{4}$, hence encoder produces 0.

Hence, $||\hat{x} - x||_2 = ||Wz\ - x\ ||_2 = \epsilon$, which is small!

Good reconstruction!!

Variants, variants, variants

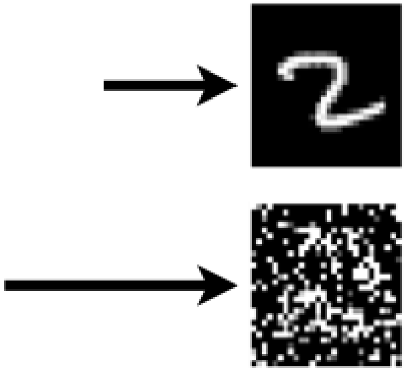
Undercomplete Representation

Hidden layer is *undercomplete* if smaller than the input layer (bottleneck layer, e.g. dimensionality reduction):

- hidden layer "compresses" the input
- will compress well only for the training distribution (maybe not even)

Hidden units will be

- good features for the training distribution (potentially…)
- will not be robust to other types of input (not trained to compress these)



Slide Credit: Hugo Larochelle

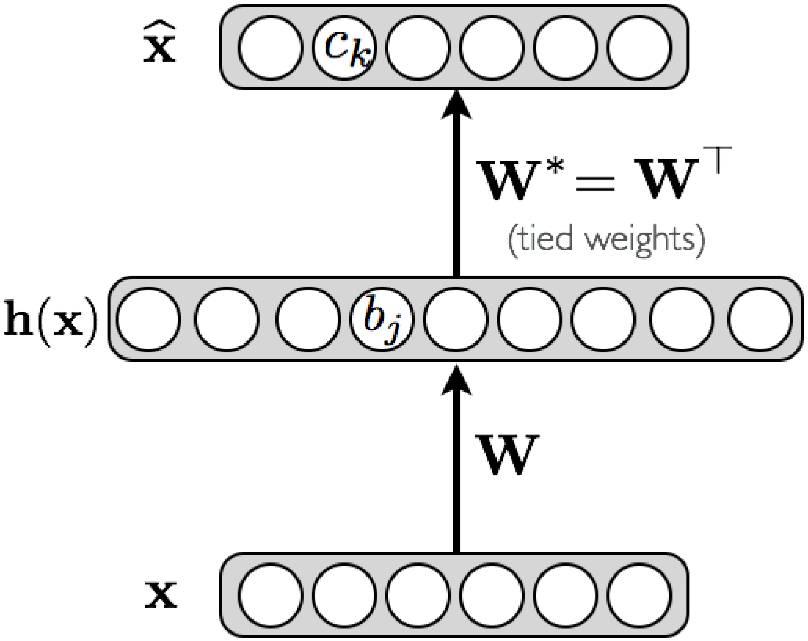
Overcomplete Representation

Hidden layer is *overcomplete* if greater than the input layer

- ➢ no compression in hidden layer
- ➢ each hidden unit could copy a different input component

No guarantee that the hidden units will extract meaningful structure

Other constraints must be made, e.g. sparsity, denoising, etc.



Slide Credit: Hugo Larochelle

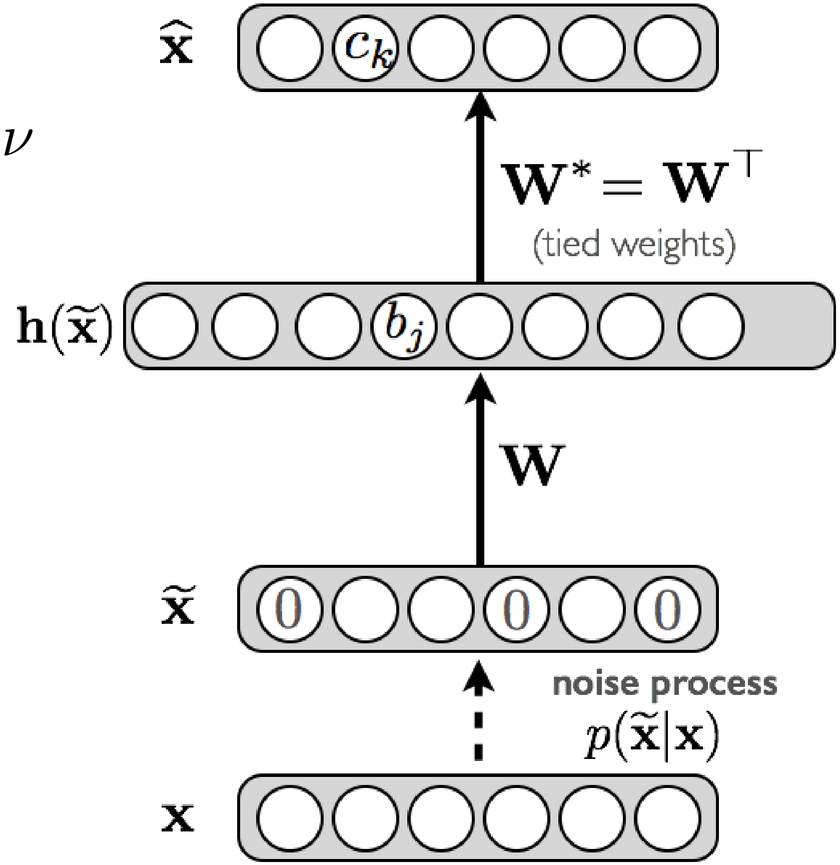
Denoising Autoencoder

Idea: representation should be *robust to introduction of noise*:

- *Dropout*: random assignment of subset of inputs to 0, with probability ν
- *Gaussian additive* noise

- Reconstruction $\widehat{\mathbf{x}}$ computed from the corrupted input $\widetilde{\mathbf{x}}$

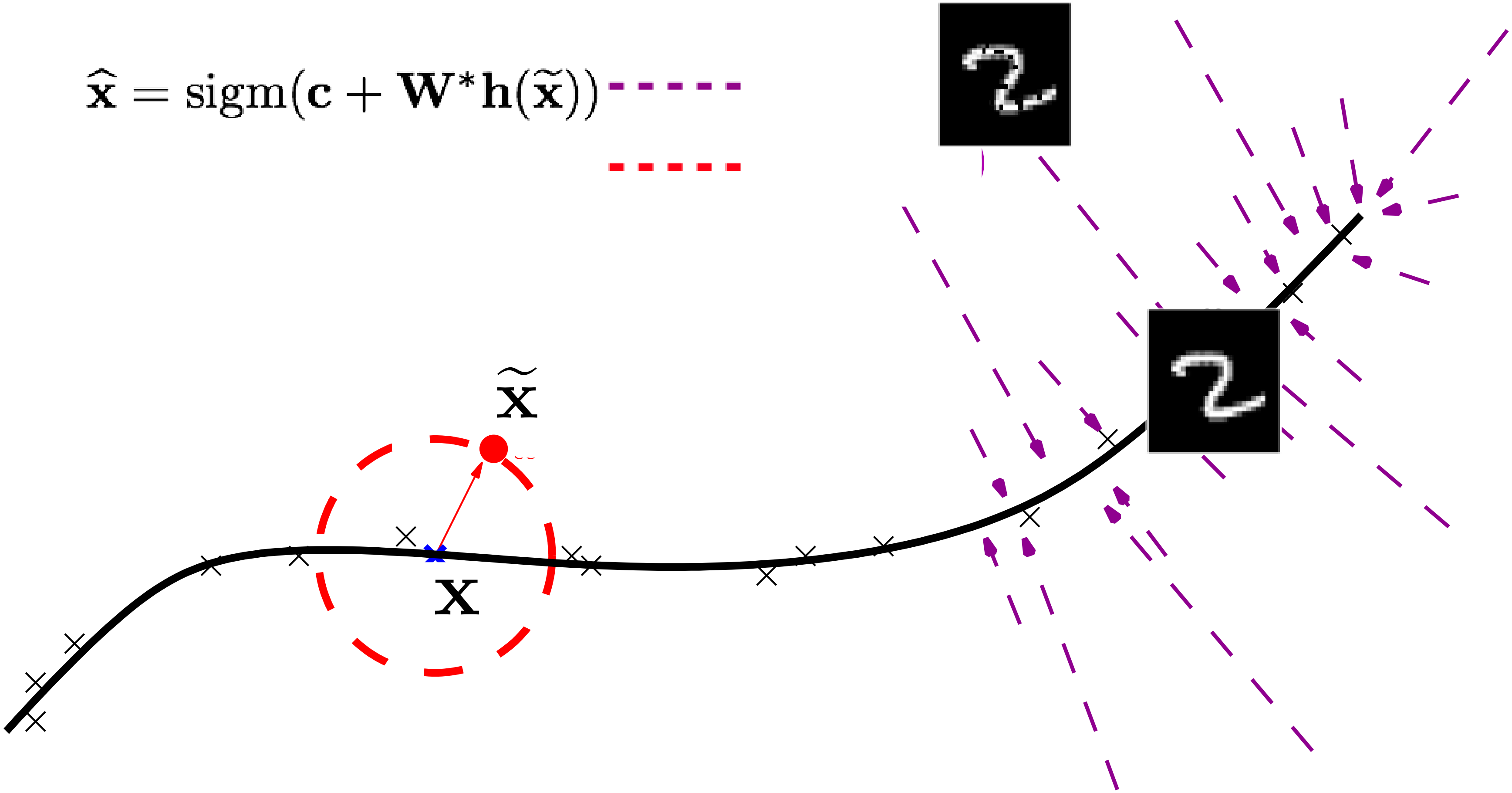
- Loss function compares $\widehat{\mathbf{x}}$ reconstruction with the noiseless input $\mathbf{x}$



Slide Credit: Hugo Larochelle

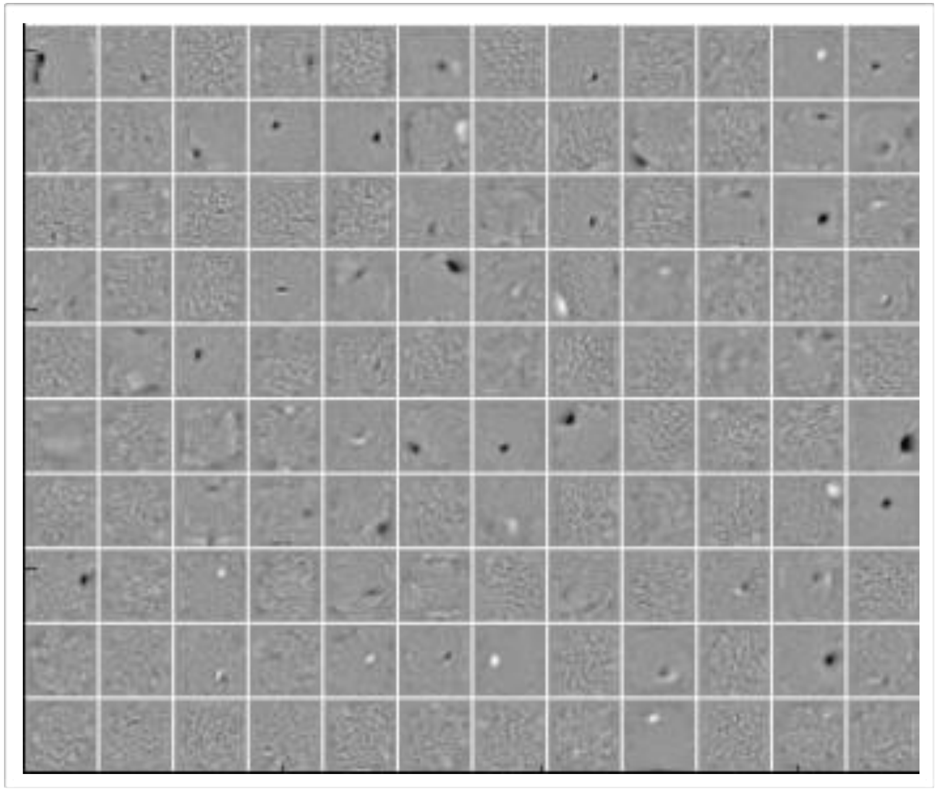
Denoising Autoencoder

$$\hat{\mathbf{x}} = \mathrm{sigm}(\mathbf{c} + \mathbf{W}^*\mathbf{h}(\tilde{\mathbf{x}}))$$

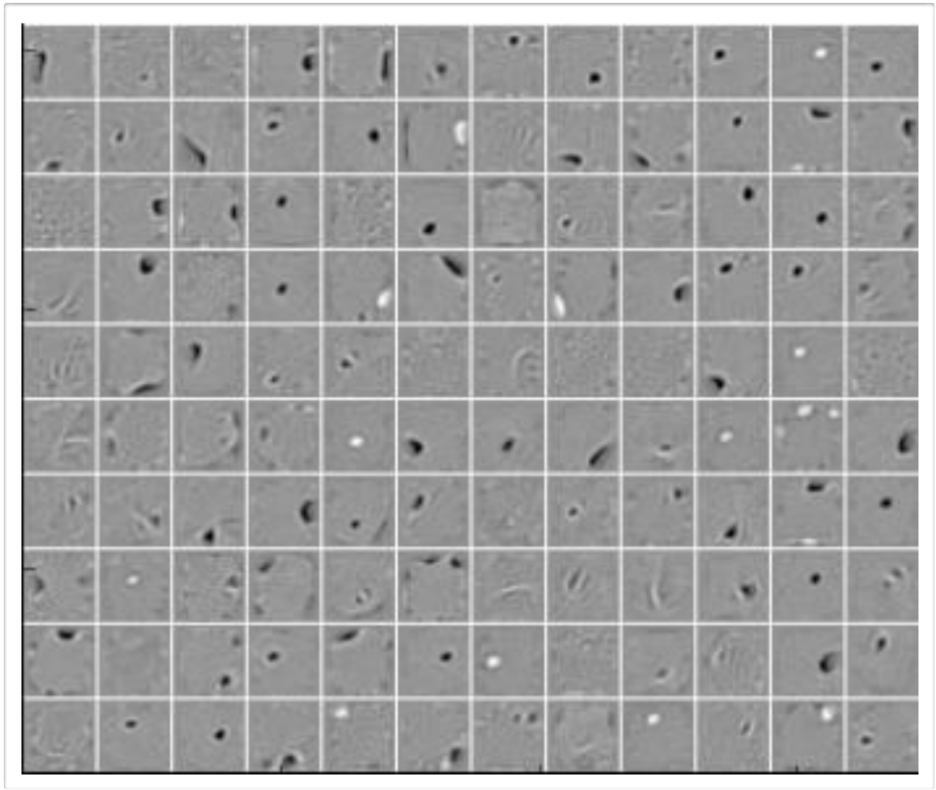


Learned Filters

Non-corrupted



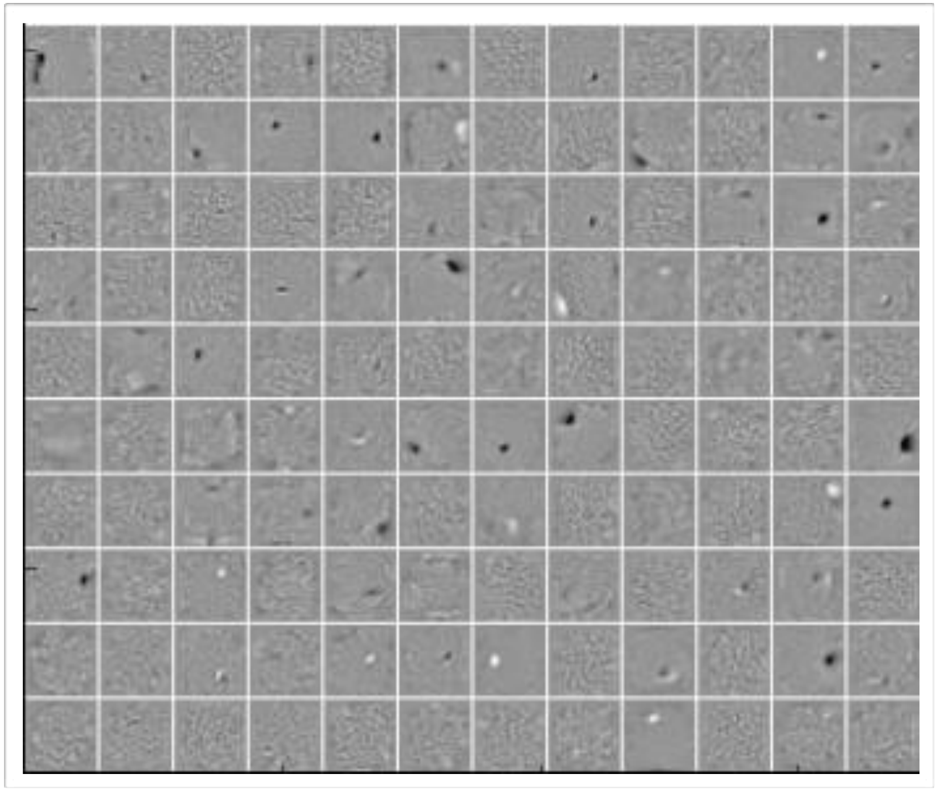
25% corrupted input



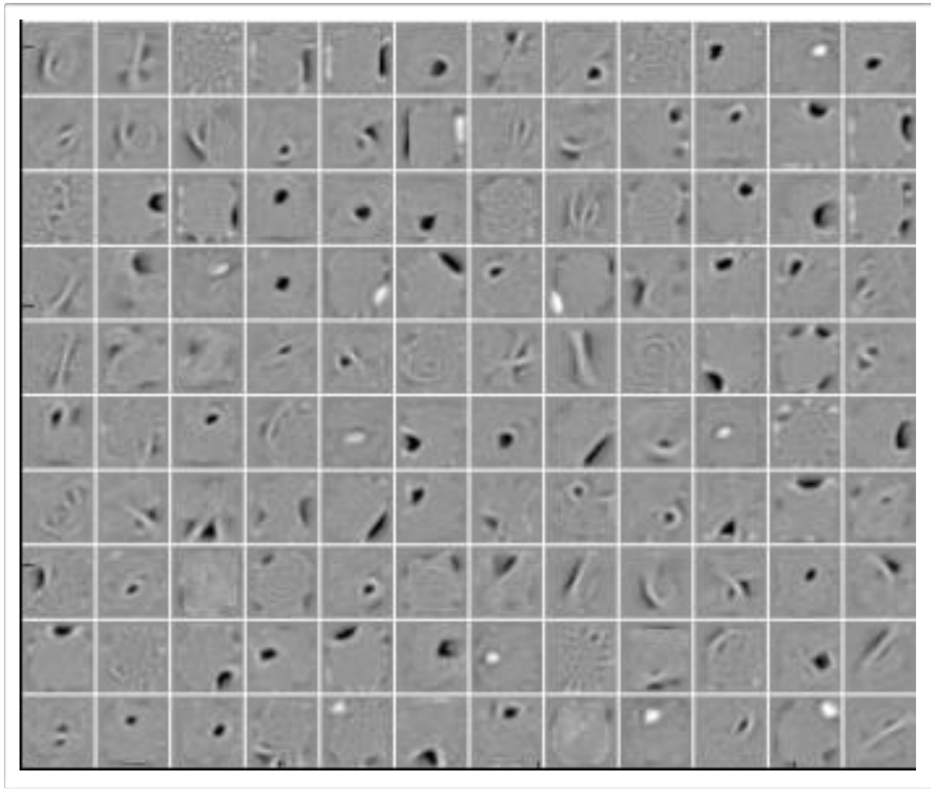
(Vincent et.al., ICML 2008)

Learned Filters

Non-corrupted



50% corrupted input

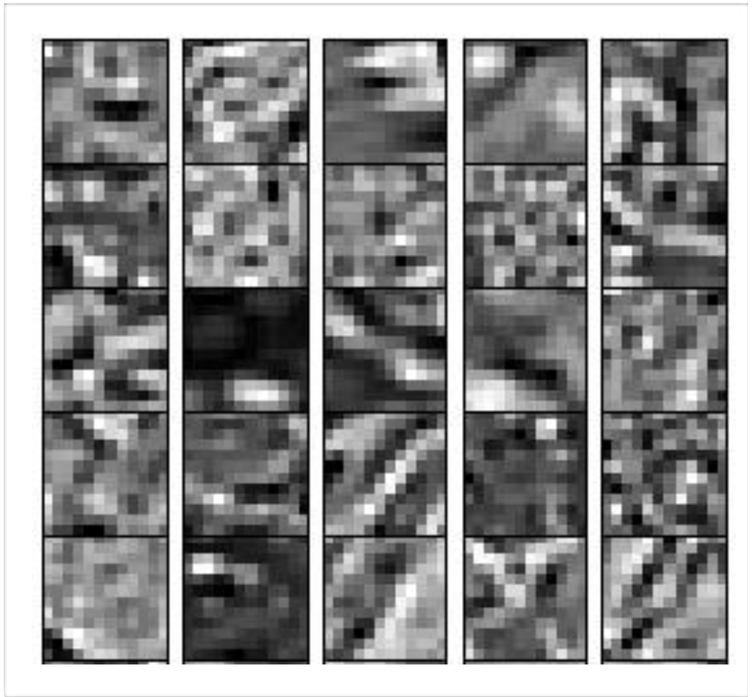


(Vincent et.al., ICML 2008)

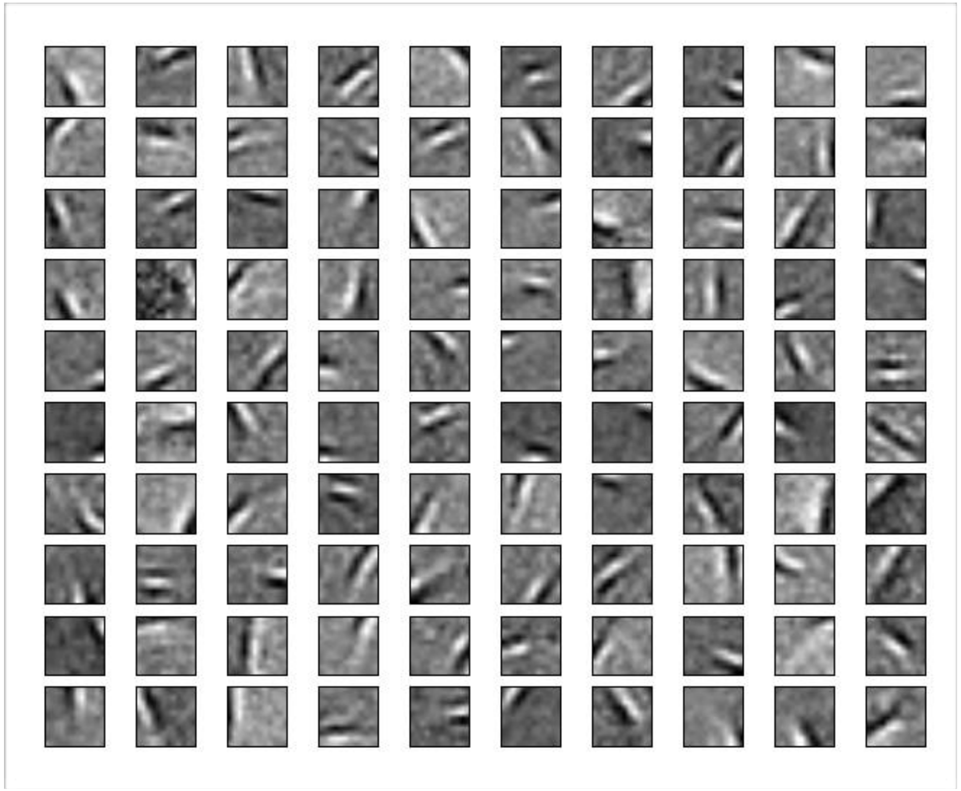
Squared Error Loss

Training on natural image patches, with squared loss

PCA may not the best solution

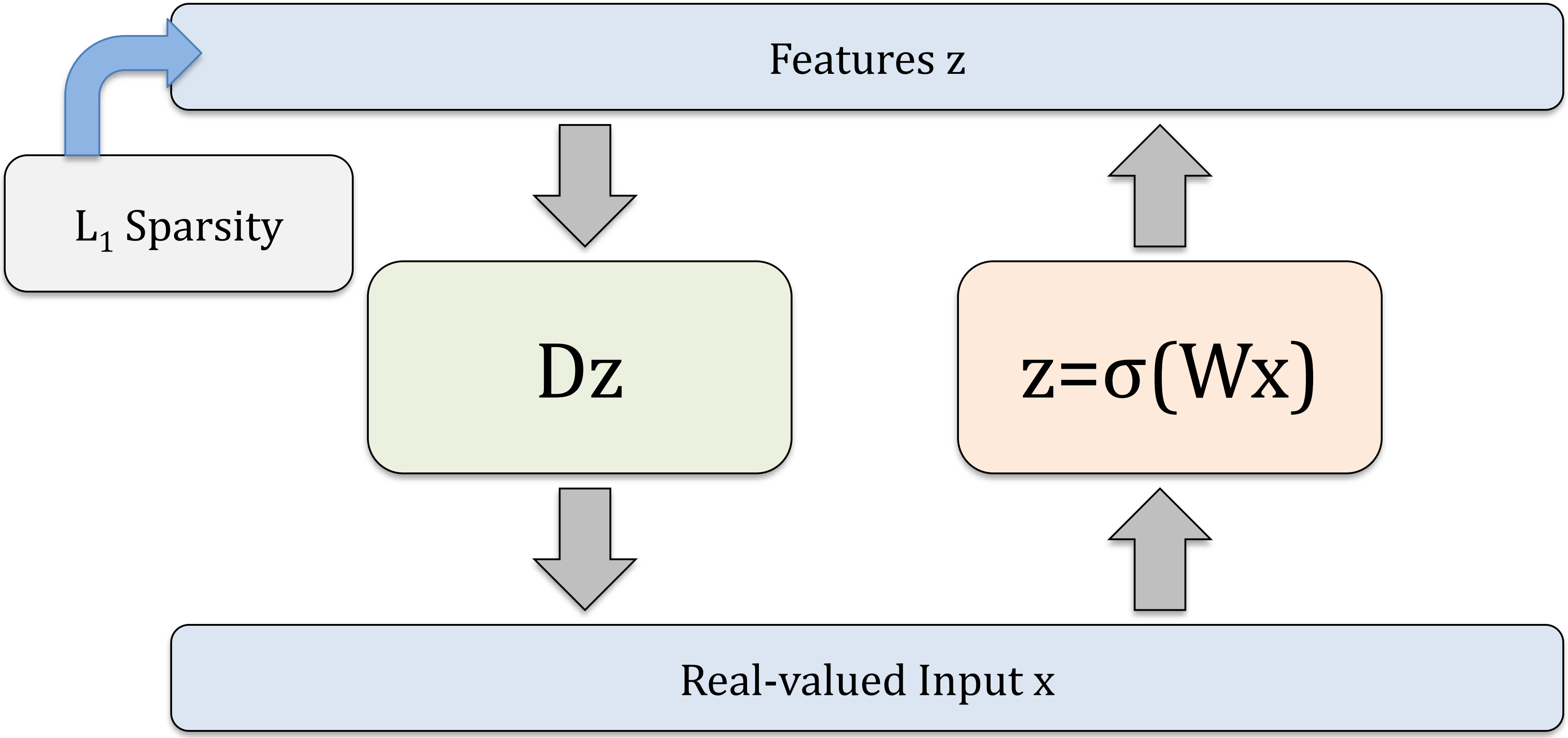


Data



Filters

Sparsity



Features z

L_1 Sparsity

Dz

z=σ(Wx)

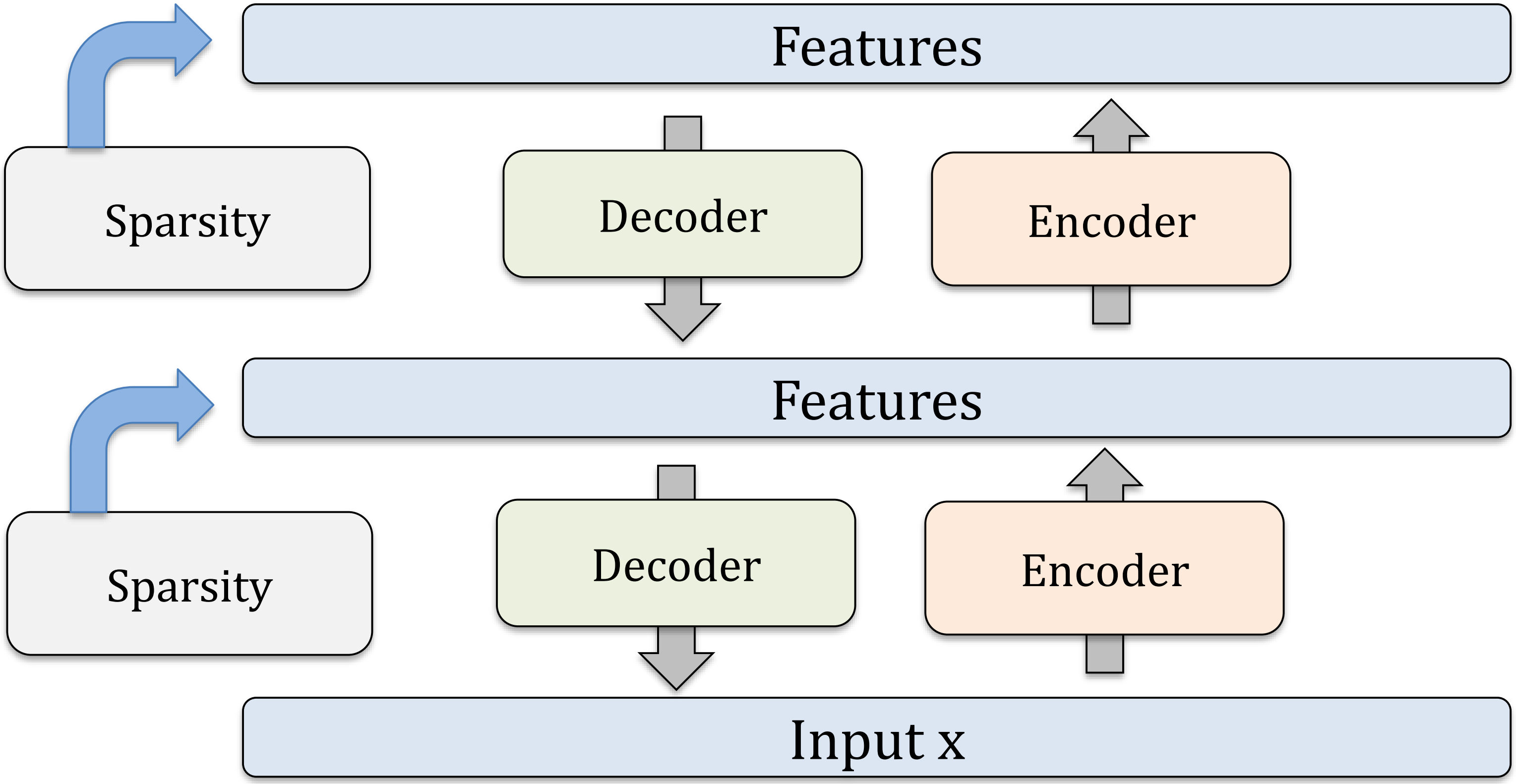
Real-valued Input x

At training time

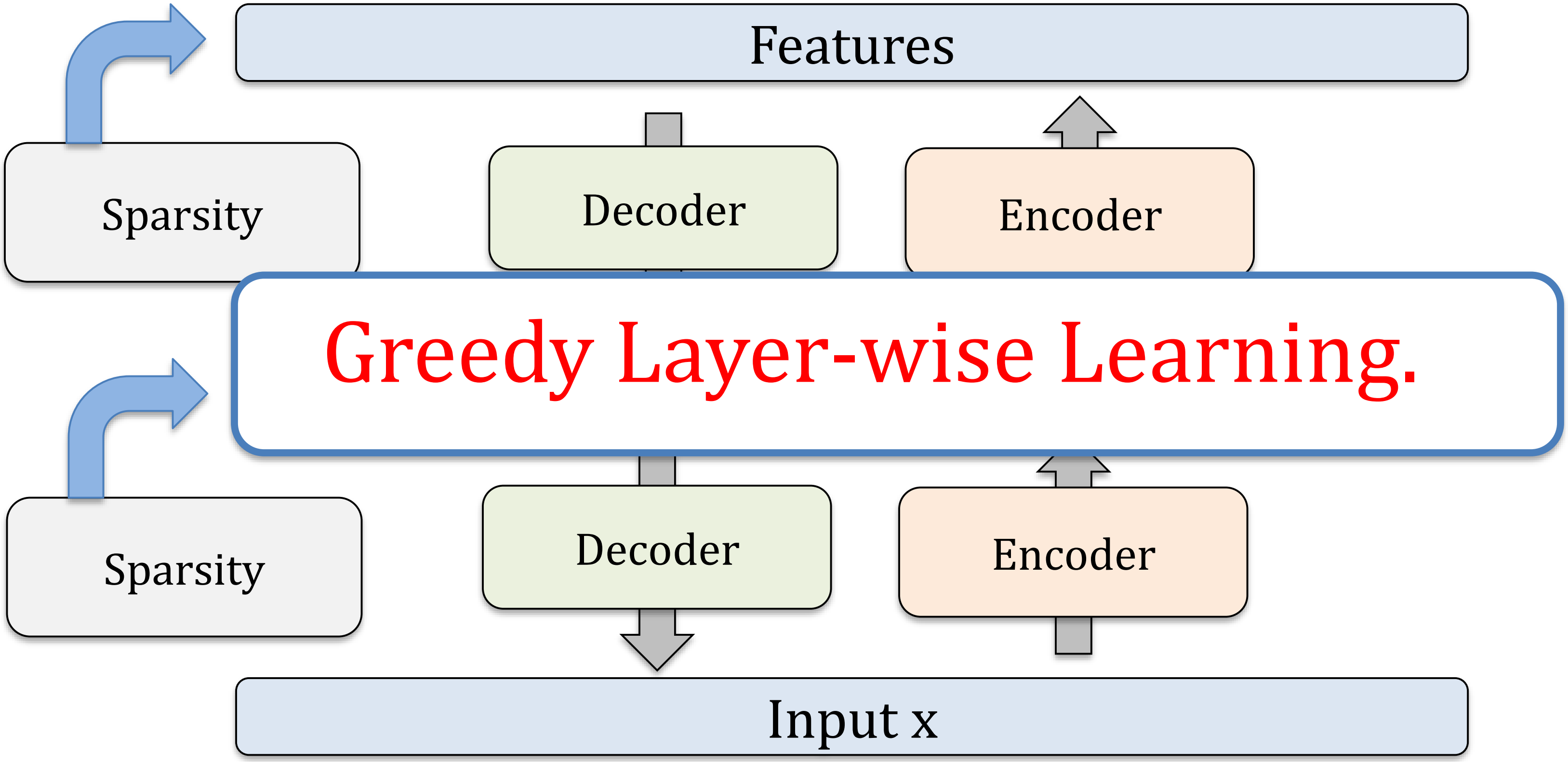
$$\min_{D,W,\mathbf{z}} \underbrace{||D\mathbf{z}-\mathbf{x}||_2^2 + \lambda|\mathbf{z}|_1}_{\text{Decoder}} + \underbrace{||\sigma(W\mathbf{x})-\mathbf{z}||_2^2}_{\text{Encoder}}$$

Kavukcuoglu et al., '09

Stacked Autoencoders



Stacked Autoencoders



Features

Sparsity

Decoder

Encoder

Greedy Layer-wise Learning.

Sparsity

Decoder

Encoder

Input x

Parameters can be fine-tuned using backpropagation.