

CIFAR
CANADIAN
INSTITUTE
FOR
ADVANCED
RESEARCH



Deep Learning Summer School 2015

On manifolds and autoencoders

by **Pascal Vincent**



Montreal Institute for Learning Algorithms

August 5, 2015

Université
de Montréal

Département d'informatique et de recherche
opérationnelle

PLAN

Part I: Leveraging the manifold hypothesis

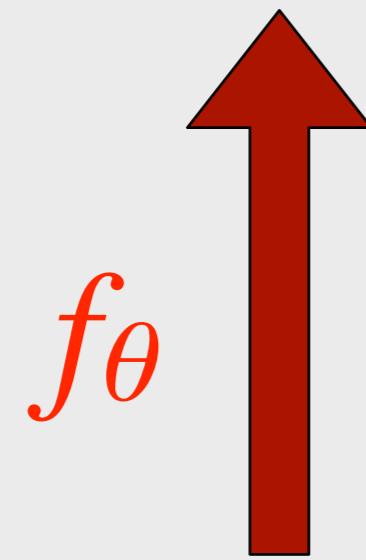
Part II: Regularizing Auto-Encoders

Will be largely about
unsupervised learning

An unsupervised learning task: dimensionality reduction

$$\mathbf{h} \in \mathbb{R}^M \quad M < D$$

$$(0.32, -1.3, 1.2)$$



$$(3.5, -1.7, 2.8, -3, 5, -1.4, 2.4, 2.7, 7.5)$$

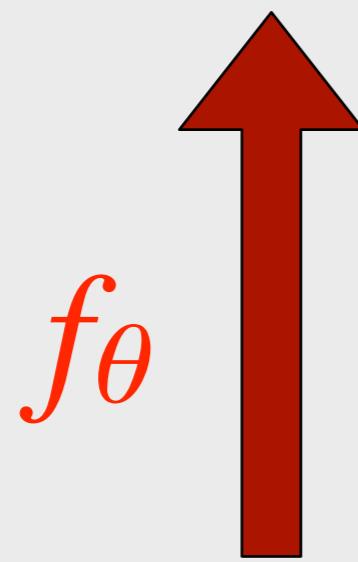
$$\mathbf{x} \in \mathbb{R}^D$$

What is it useful for?

An unsupervised learning task: dimensionality reduction

$$\mathbf{h} \in \mathbb{R}^M \quad M < D$$

(0.32, -1.3, 1.2)



(3.5, -1.7, 2.8, -3, 5, -1.4, 2.4, 2.7, 7.5)

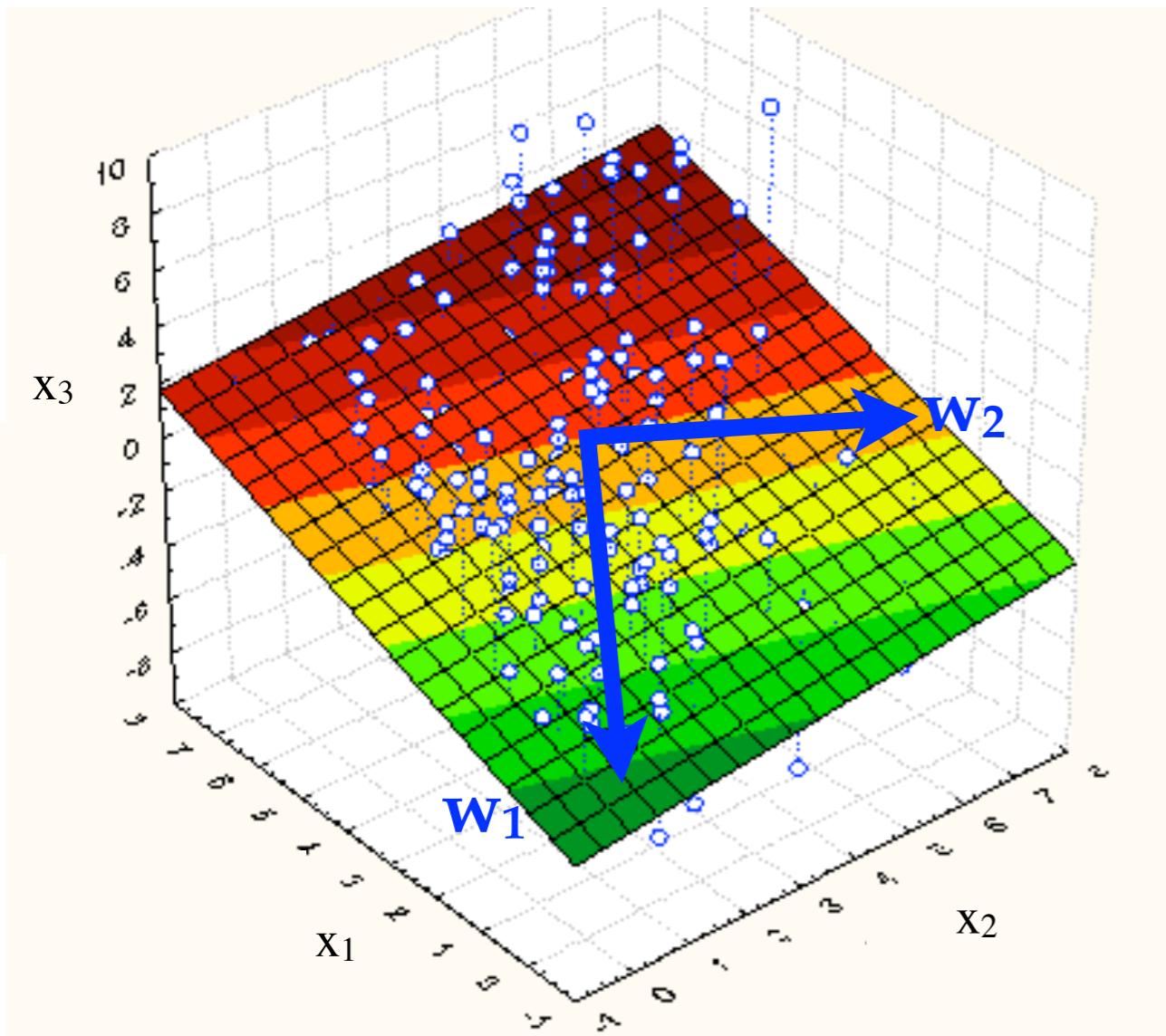
$$\mathbf{x} \in \mathbb{R}^D$$

What is it useful for?

- Data compression (lossy)
- Dataset visualisation (in 2D or 3D)
- Discovering «most important» features.

A classic algorithm [Pearson 1901] [Hotelling 1933]

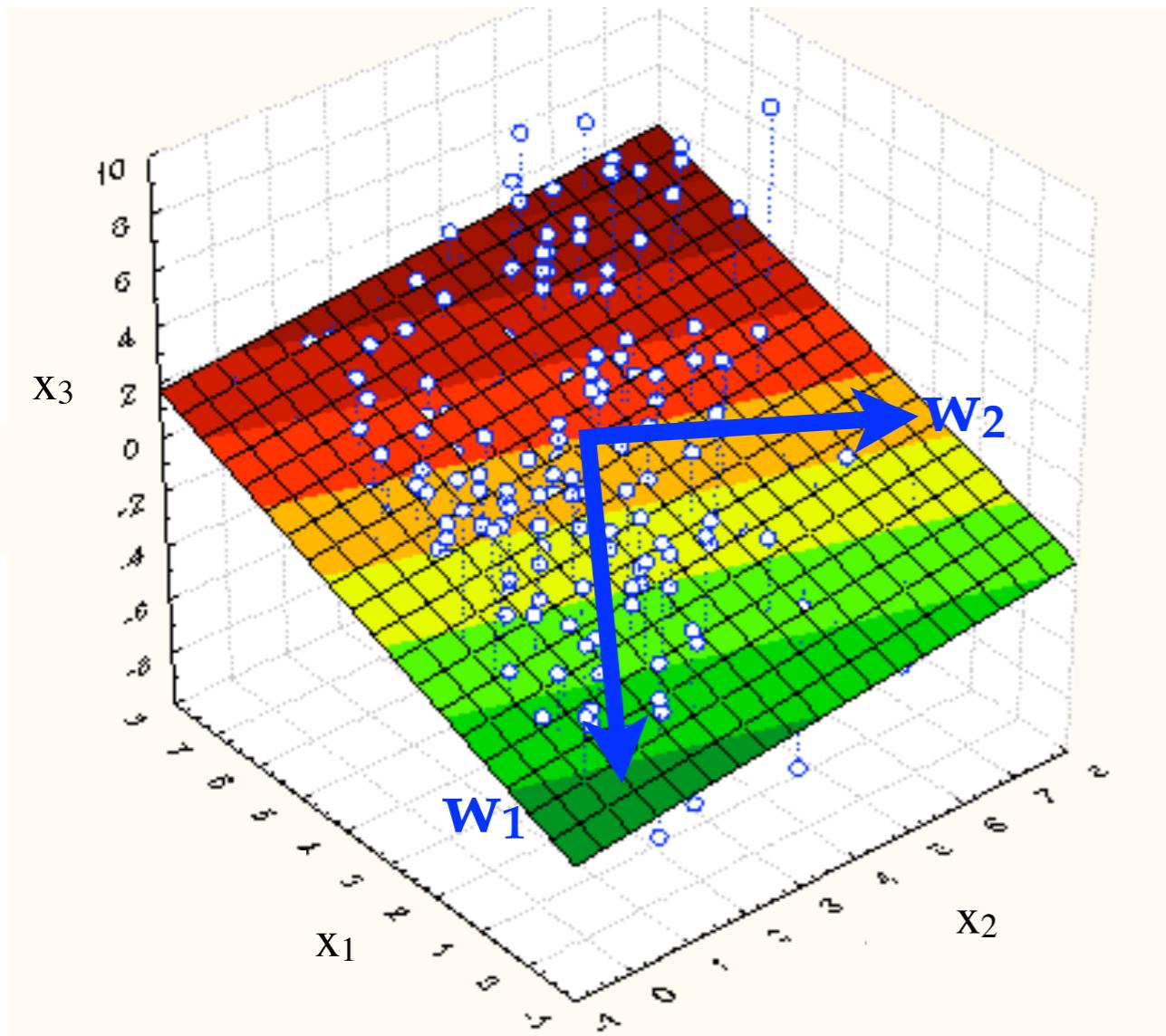
Principal Component Analysis



- Finds (learns) k directions (a subspace) in which data has highest variance
=> *principal directions* (eigenvectors) W
- Projecting inputs x on these vectors yields reduced dimension representation (&decorrelated)
=> *principal components*
 $h = f_\theta(x) = W(x - \mu)$ with $\theta = \{W, \mu\}$

A classic algorithm [Pearson 1901] [Hotelling 1933]

Principal Component Analysis



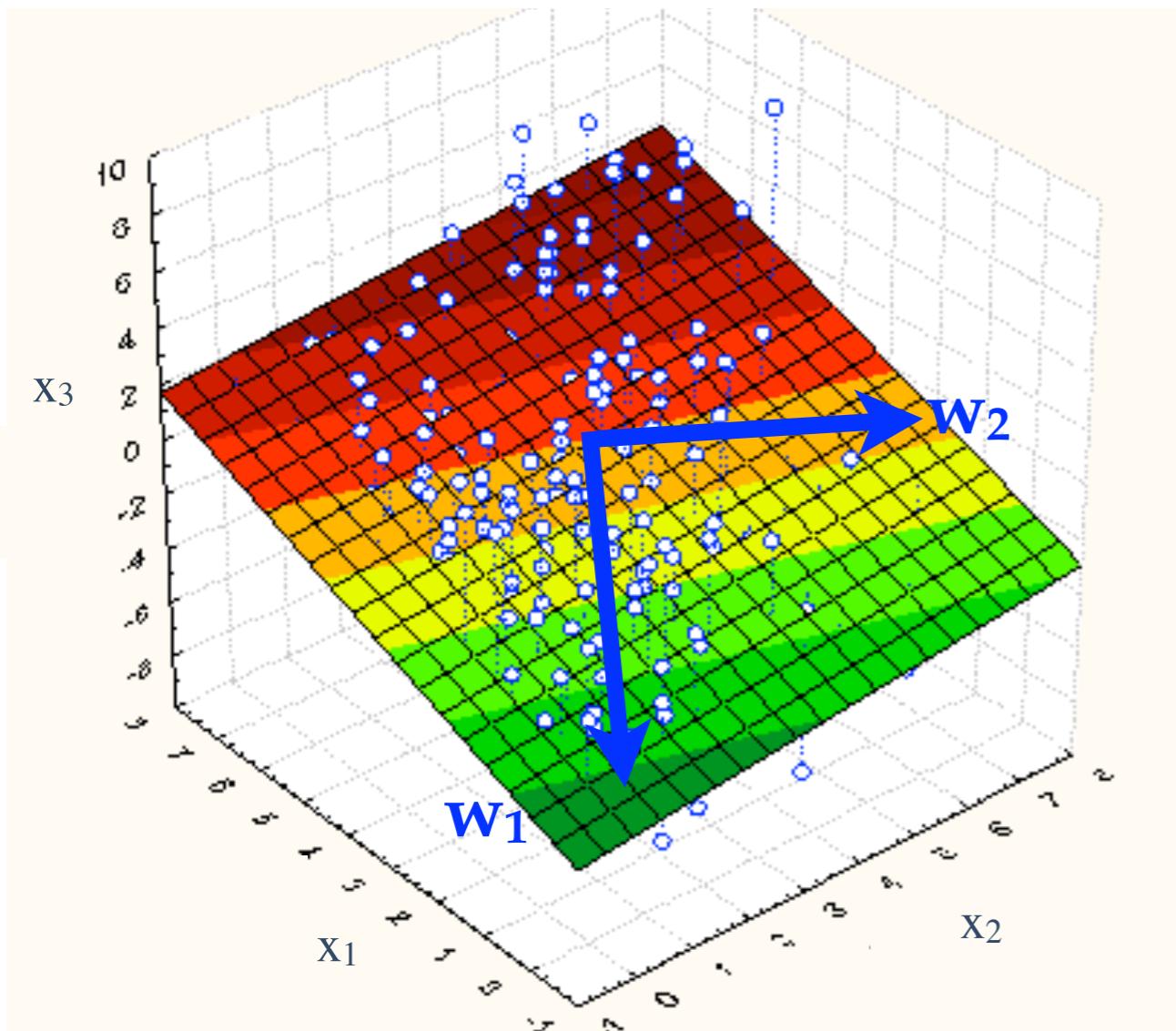
- Finds (learns) k directions (a subspace) in which data has highest variance
=> *principal directions* (eigenvectors) W
- Projecting inputs x on these vectors yields reduced dimension representation (&decorrelated)
=> *principal components*
 $h = f_\theta(x) = W(x - \mu)$ with $\theta = \{W, \mu\}$

Why mention PCA?

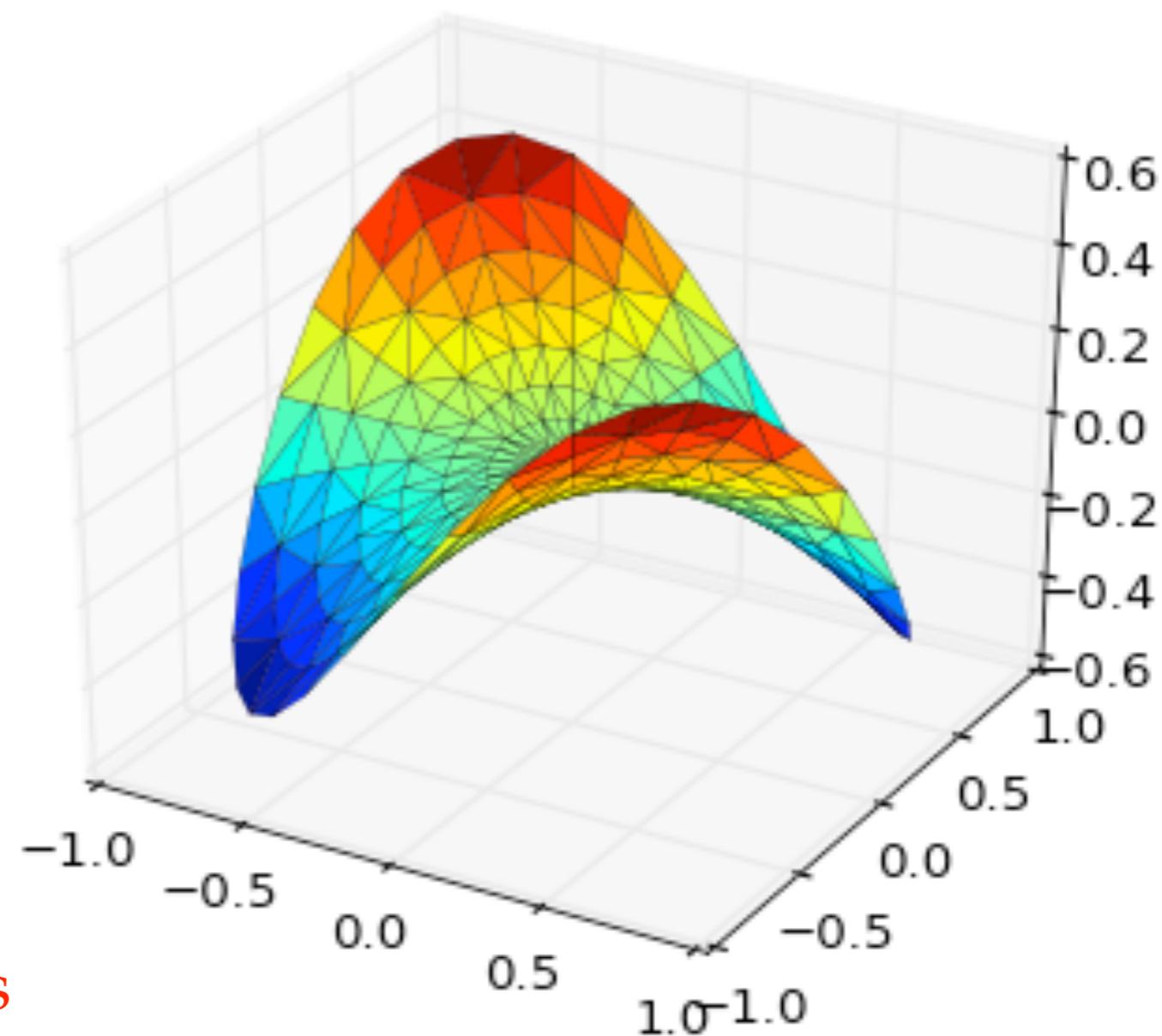
- Prototypical unsupervised representation learning algorithm.
- Related to autoencoders
- Prototypical manifold modeling algorithm

Lower-dimensional manifolds embedded in high dimensional space

Linear 2D manifold in 3D space
(ex: subspace found by PCA)



Non-linear 2D manifold
in 3D input space



Principal components are coordinates
in a coordinate-system on the manifold

The manifold hypothesis (assumption)

**Natural data in high dimensional spaces
concentrates
close to lower dimensional manifolds.**

Probability density decreases very rapidly when moving away from the supporting manifold.

The curse of dimensionality

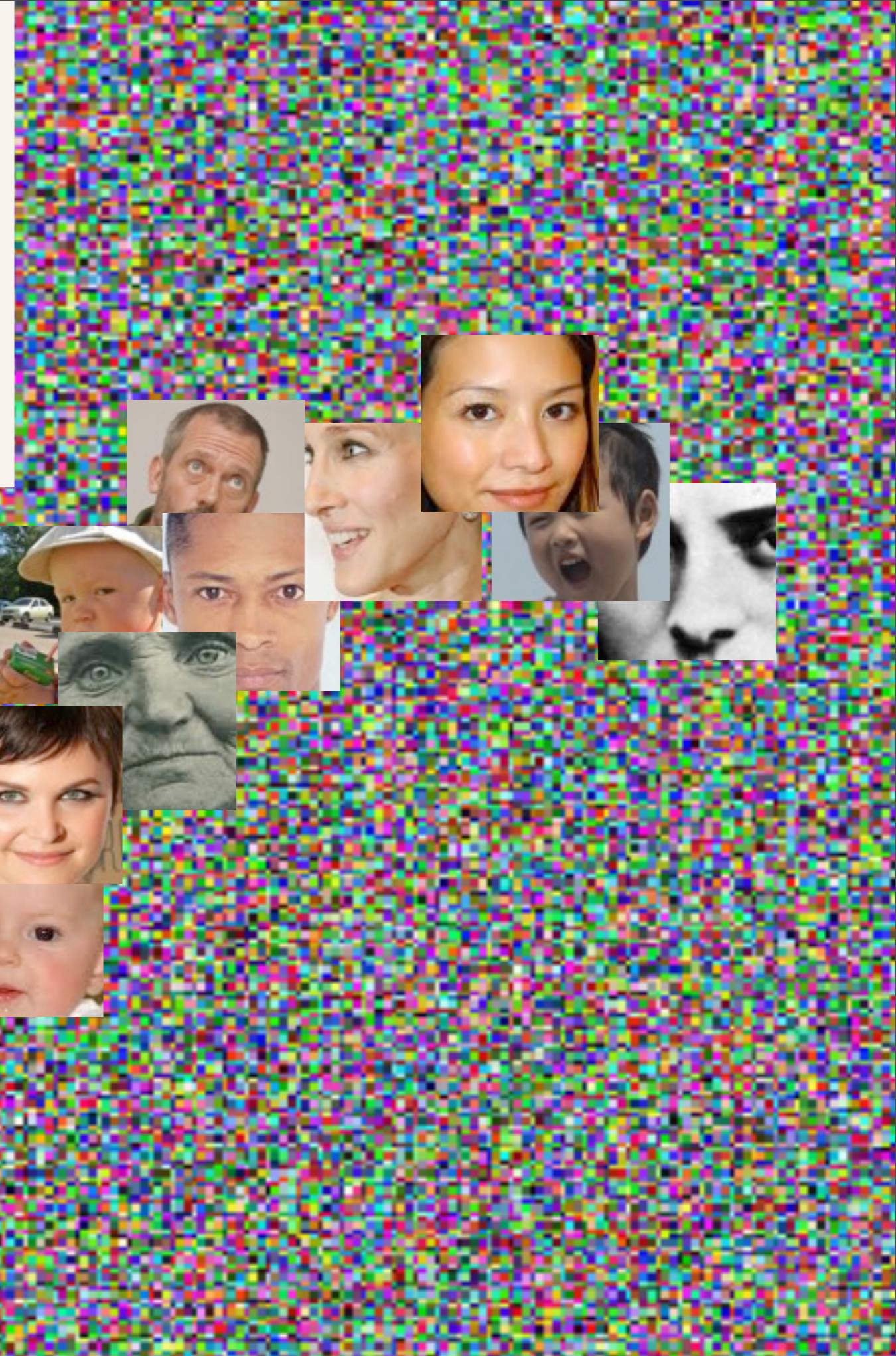
There are 10^{96329} possible 200x200 RGB images.



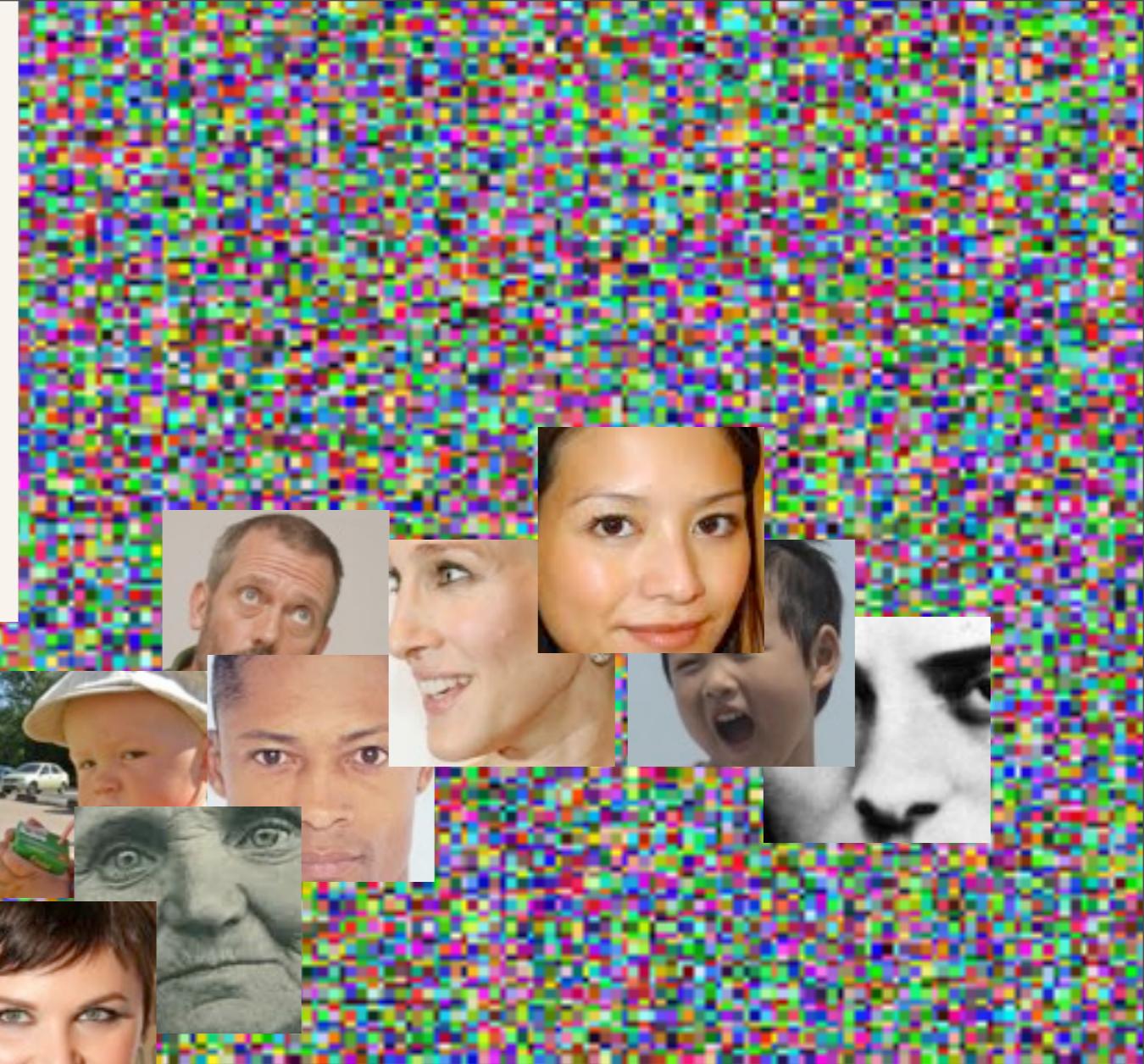




- Natural images occupy a tiny fraction of that space
=> suggests peaked density
- Realistic smooth transformations from one image to another
=> continuous path along manifold



- Natural images occupy a tiny fraction of that space
=> suggests peaked density
- Realistic smooth transformations from one image to another
=> continuous path along manifold



The manifold hypothesis

Data density concentrates near a lower dimensional manifold



Can shift the curse
from high d to $d_M \ll d$

Manifold follows naturally from continuous underlying factors (\approx intrinsic manifold coordinates)

Ex: pose parameters of a face

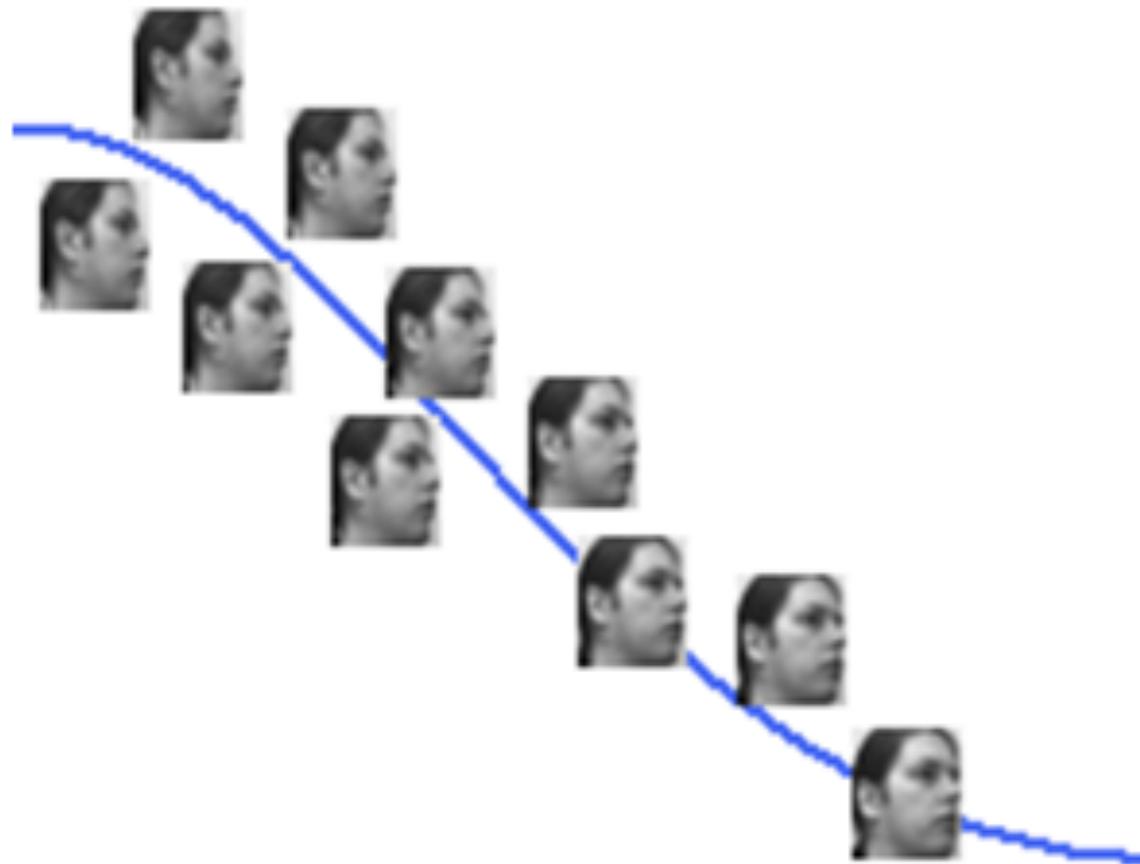
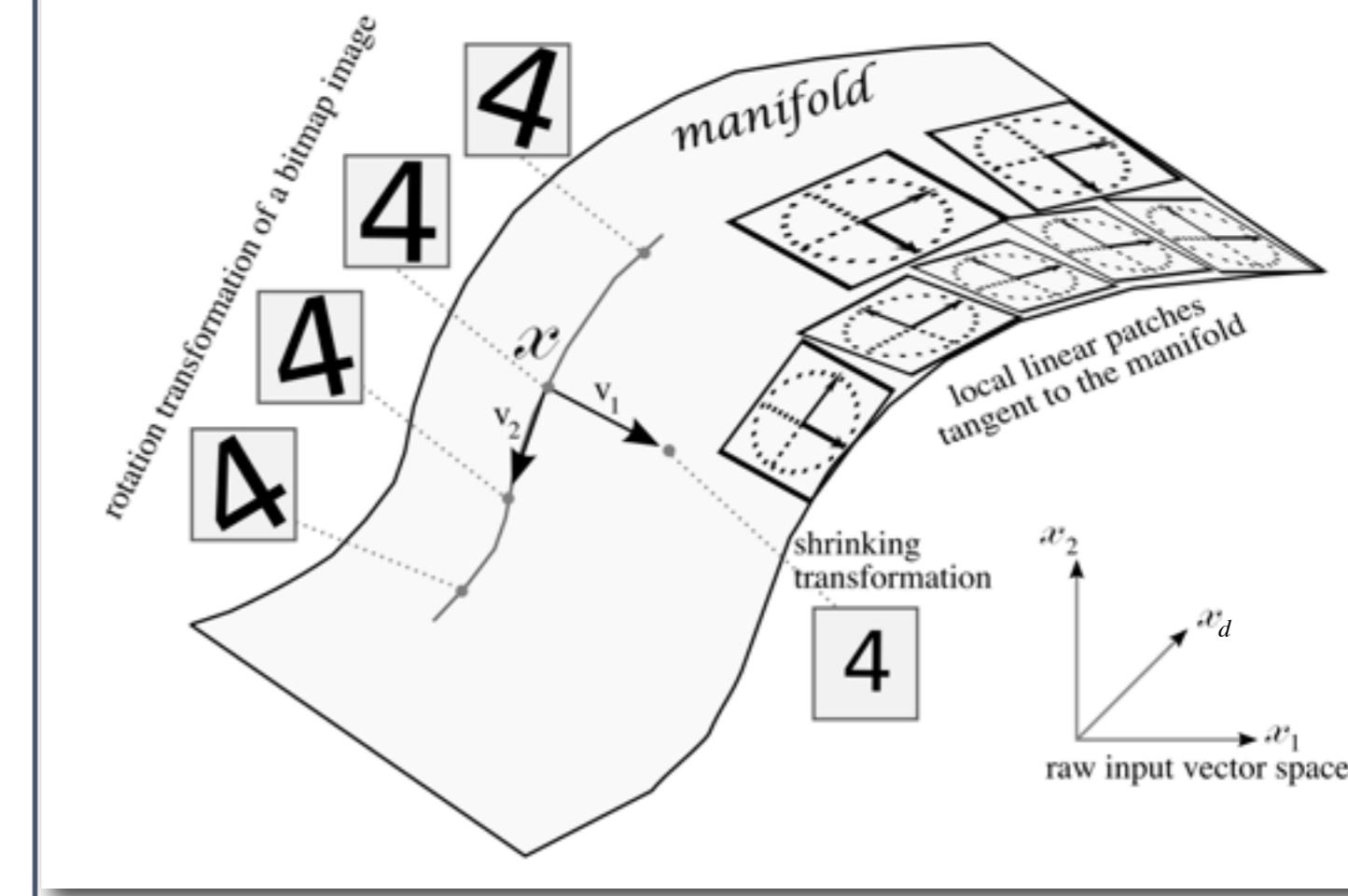


Image borrowed from University of Dayton Vision Lab website.

Ex: rotation, size of digits (+ line thickness, ...)

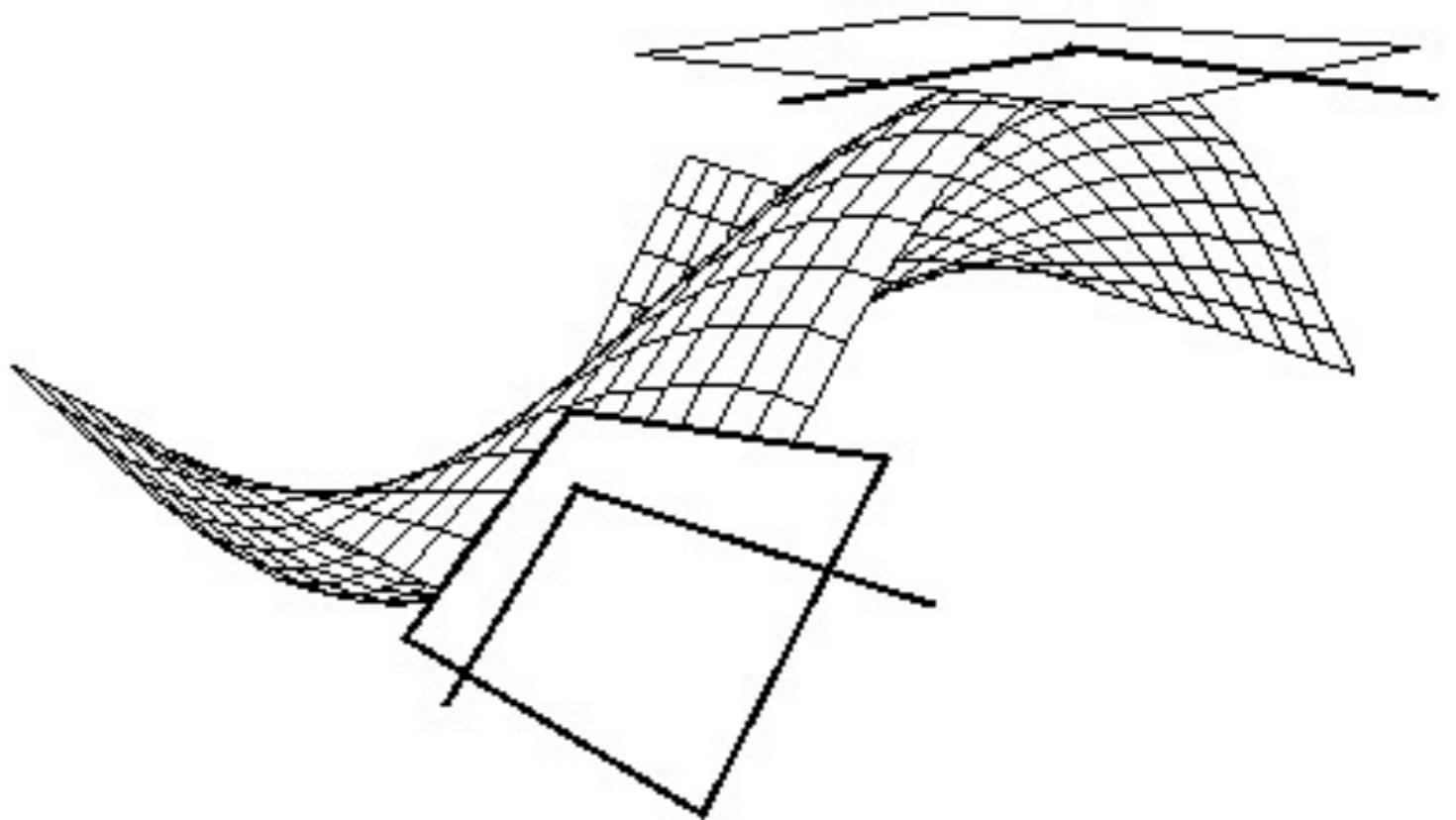


Such continuous factors are
(part of) a **meaningful representation!**

Modeling local tangent spaces

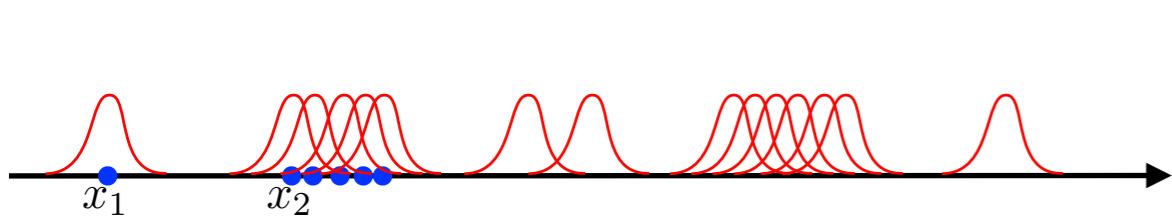
A non-linear manifold

- Can be represented by **patchwork** of **tangent spaces**
- Yields *local* linear coordinate systems (chart \rightarrow atlas)



Non-parametric density estimation

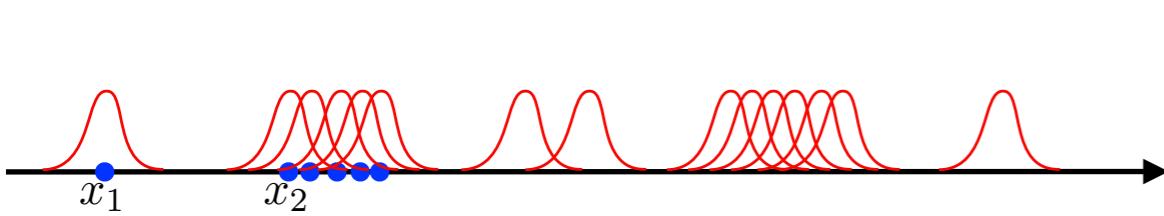
Non-parametric density estimation



$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; x_i, C_i)$$

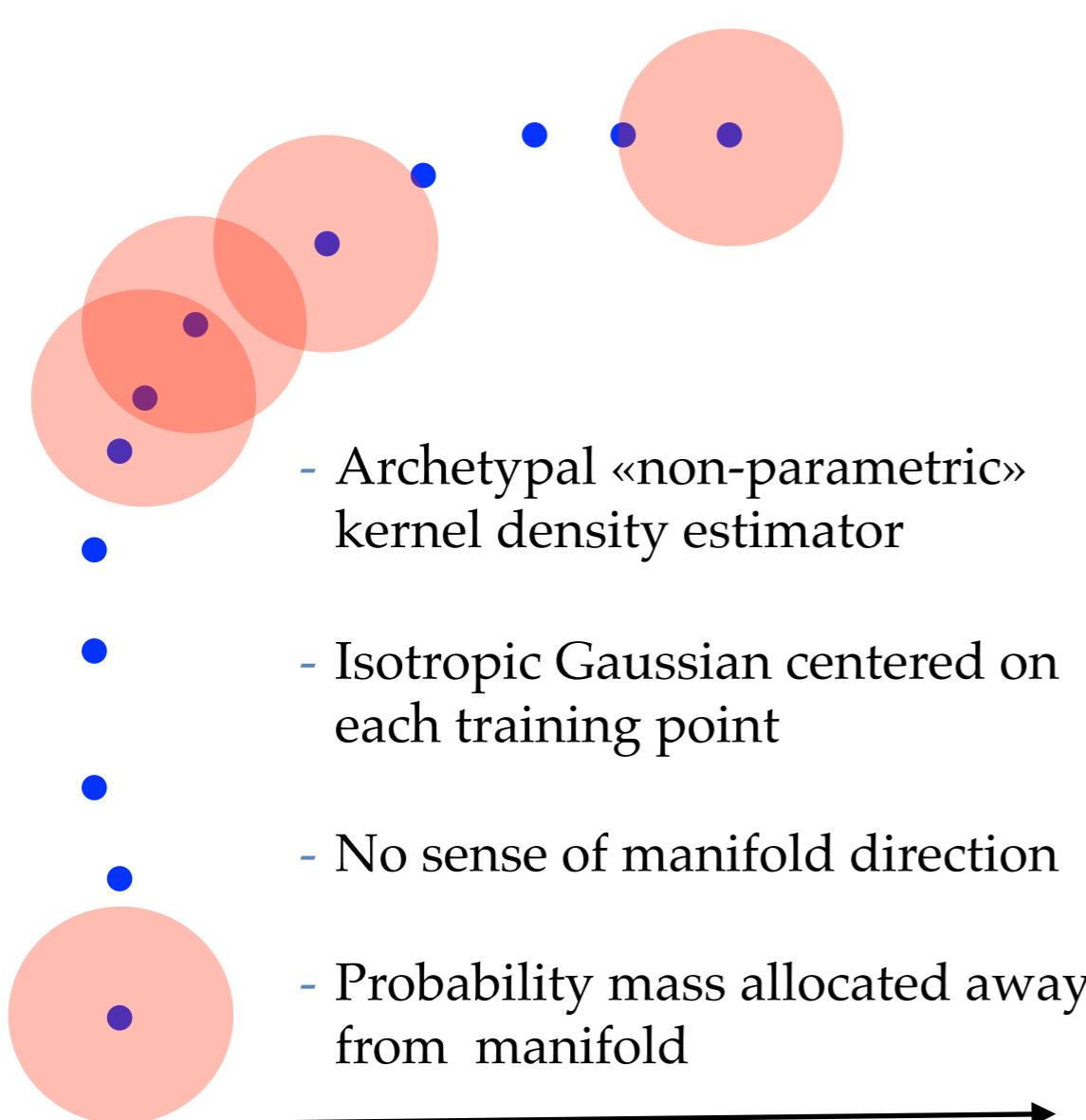
Classical Parzen Windows
density estimator

Non-parametric density estimation

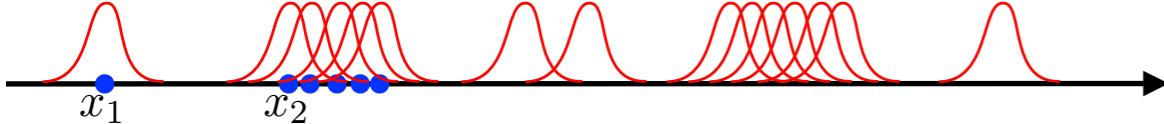


$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; x_i, C_i)$$

Classical Parzen Windows density estimator

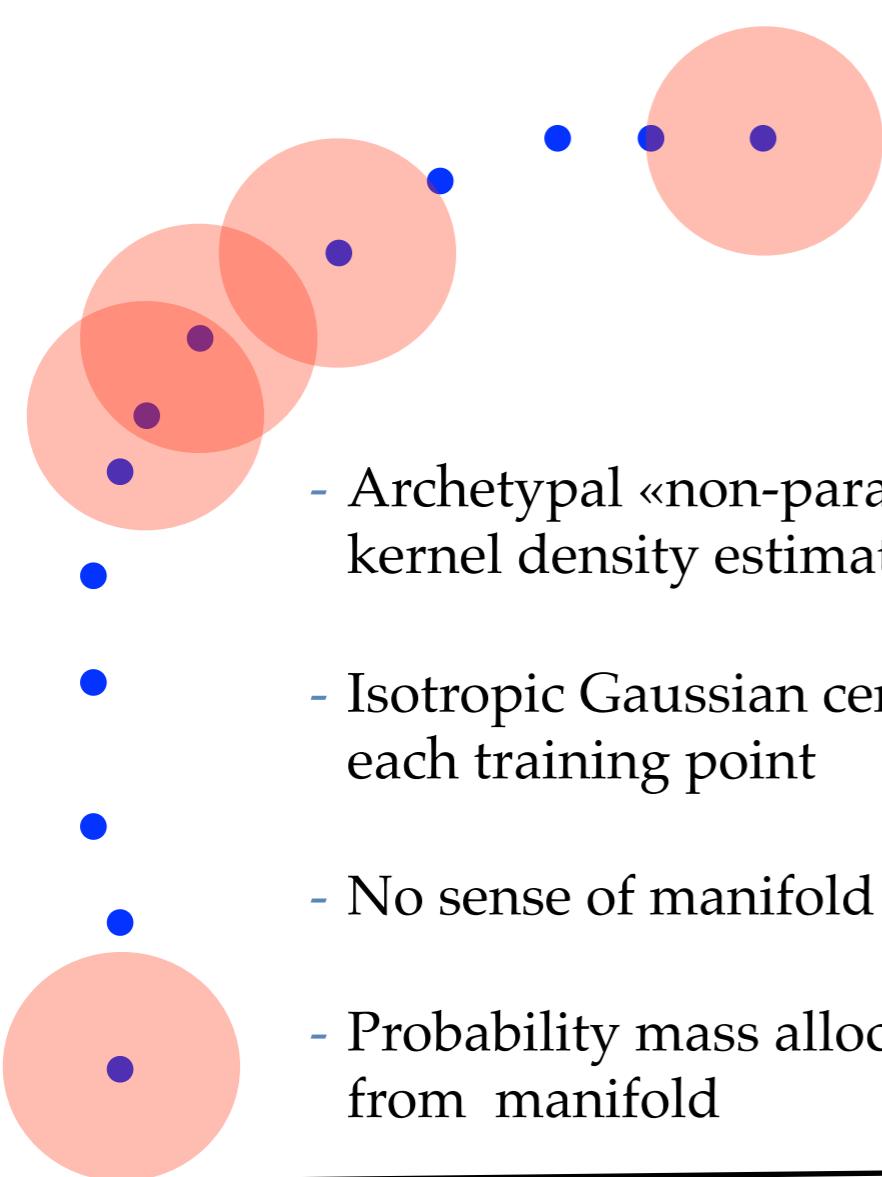


Non-parametric density estimation



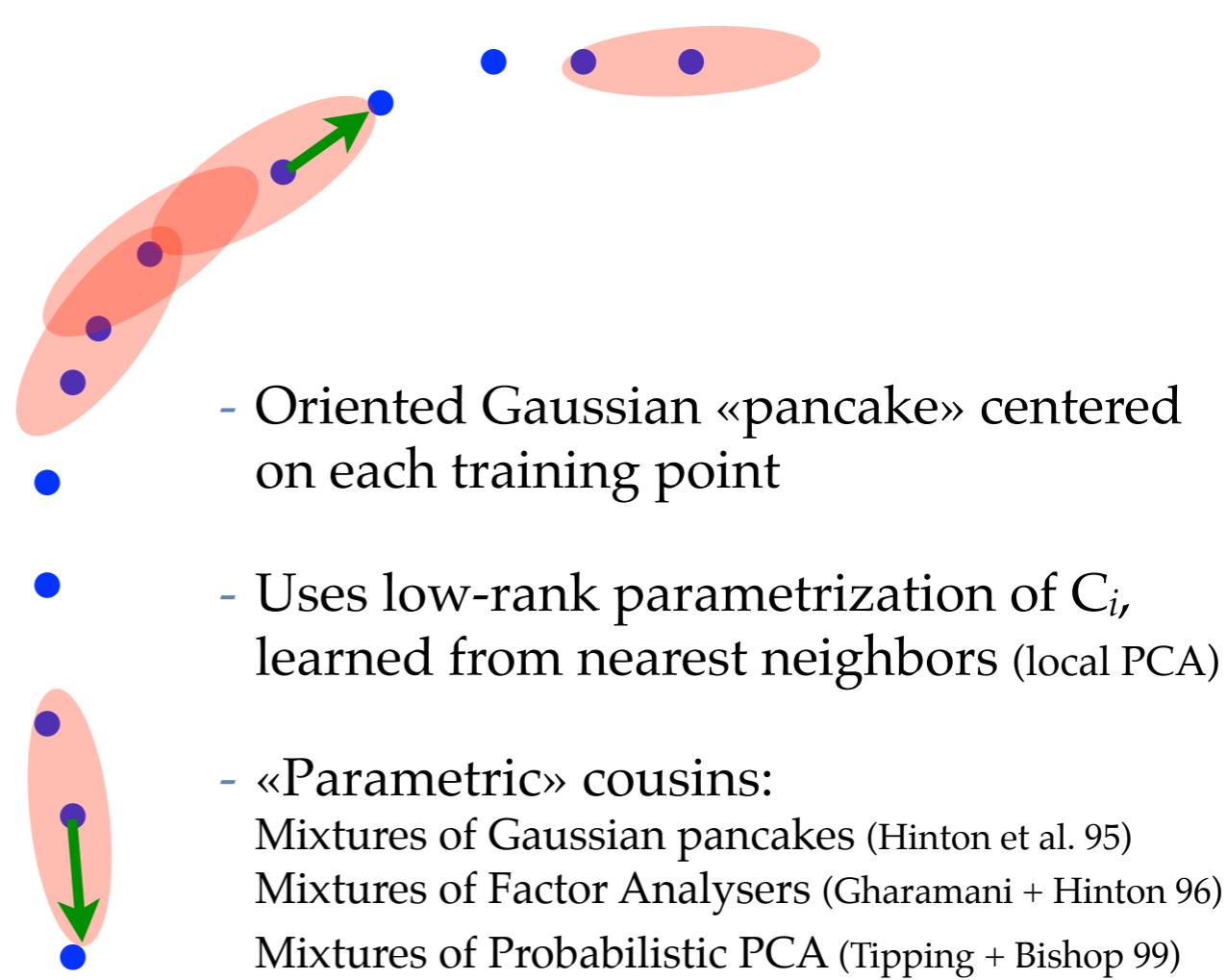
$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; x_i, C_i)$$

Classical Parzen Windows
density estimator



Manifold Parzen Windows
density estimator

(Vincent and Bengio, NIPS 2003)



Non-local manifold Parzen windows

(Bengio, Larochelle, Vincent, NIPS 2006)

Isotropic Parzen:

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; x_i, \underbrace{\sigma^2 I}_{\text{isotropic}})$$

Non-local manifold Parzen windows

(Bengio, Larochelle, Vincent, NIPS 2006)

Isotropic Parzen:

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; x_i, \underbrace{\sigma^2 I}_{\text{isotropic}})$$

Manifold Parzen:

(Vincent and Bengio, NIPS 2003)

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; x_i, \underbrace{C_i}_{d_M \text{ high variance directions from PCA on } k \text{ nearest neighbors}})$$

Non-local manifold Parzen windows

(Bengio, Larochelle, Vincent, NIPS 2006)

Isotropic Parzen:

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; x_i, \underbrace{\sigma^2 I}_{\text{isotropic}})$$

Manifold Parzen:

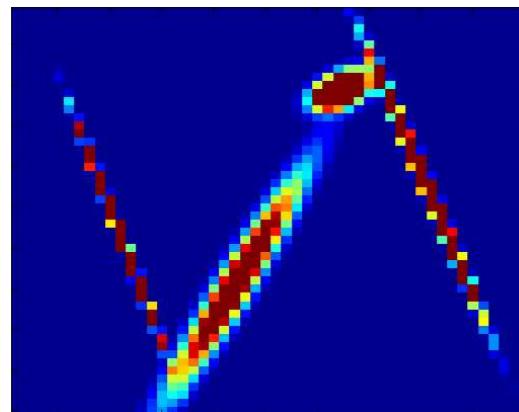
(Vincent and Bengio, NIPS 2003)

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; x_i, \underbrace{C_i}_{d_M \text{ high variance directions from PCA on } k \text{ nearest neighbors}})$$

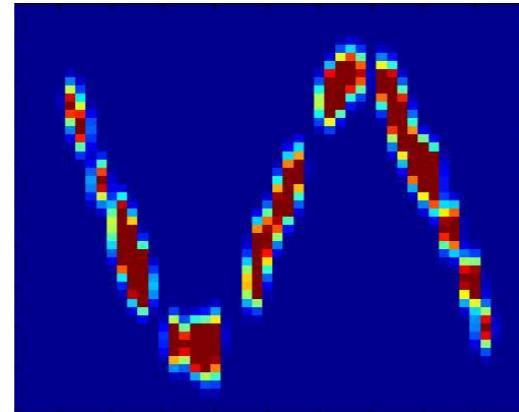
Non-local manifold Parzen:

(Bengio, Larochelle, Vincent, NIPS 2006) $\hat{p}(x) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(x; \underbrace{\mu(x_i), C(x_i)}_{d_M \text{ high variance directions output by neural network trained to maximize likelihood of } k \text{ nearest neighbors}})$

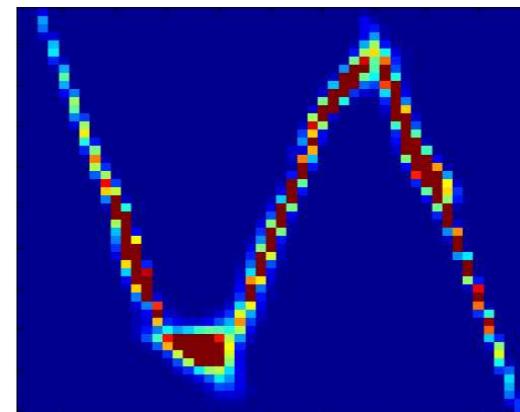
Mixture of k
Gaussians



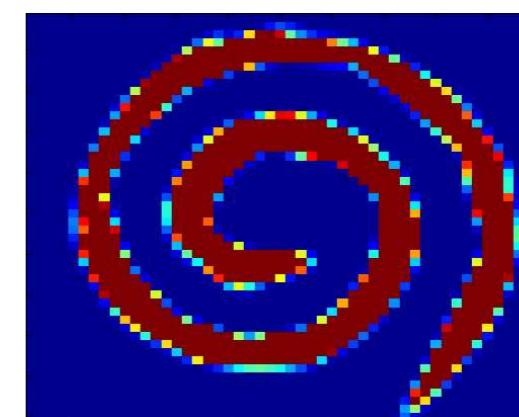
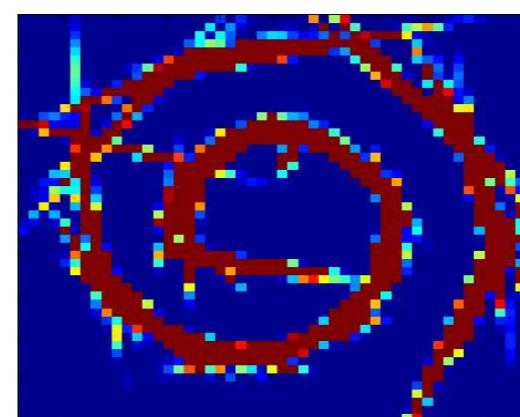
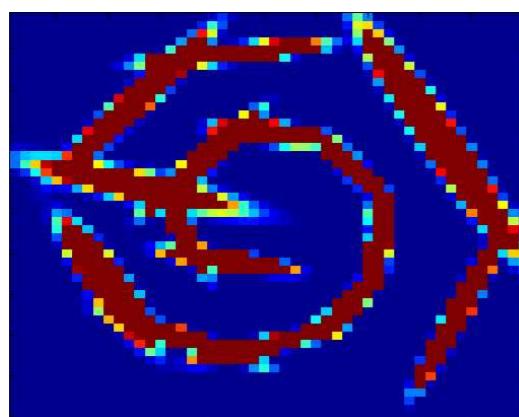
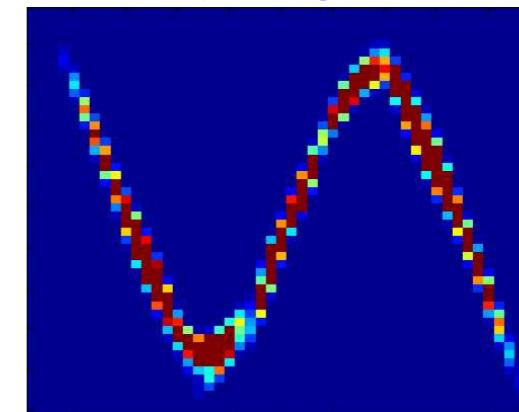
Parzen
Windows



Manifold
Parzen



Non-local
Manifold
Parzen



Use in Bayes classifier on USPS

Algorithm	Valid.	Test	Hyper-Parameters
SVM	1.2%	4.68%	$C = 100, \sigma = 8$
Parzen Windows	1.8%	5.08%	$\sigma = 0.8$
Manifold Parzen	0.9%	4.08%	$d = 11, k = 11, \sigma_0^2 = 0.1$
Non-local MP	0.6%	3.64% (-1.5218)	$d = 7, k = 10, k_\mu = 10, \sigma_0^2 = 0.05, n_{hid} = 70$
Non-local MP*	0.6%	3.54% (-1.9771)	$d = 7, k = 10, k_\mu = 4, \sigma_0^2 = 0.05, n_{hid} = 30$

Manifold learning is a rich subfield

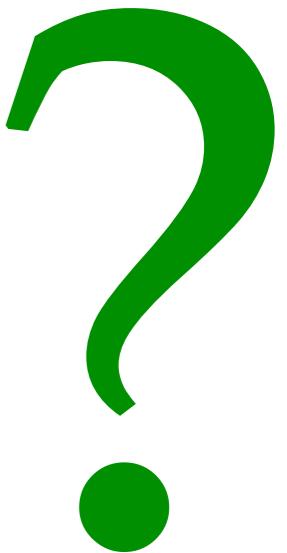
Purely non-parametric:

- Manifold Parzen, LLE, Isomap, Laplacian eigenmaps, t-SNE, ...

Learned parametrized function:

- Parametric t-SNE, semi-supervised embedding,
non-local manifold Parzen, ...

What do all these approaches
have in common



Neighborhood-based training!

- They explicitly use distance-based neighborhoods.
- Training with k-nearest neighbors, or pairs of points.
- Typically Euclidean neighbors
- But in **high d** , your nearest Euclidean neighbor can be **very** different from you...

Neighborhood-based training!

- They explicitly use distance-based neighborhoods.
- Training with k-nearest neighbors, or pairs of points.
- Typically Euclidean neighbors
- But in **high d** , your nearest Euclidean neighbor can be **very** different from you...



Neighborhood-based training!

- They explicitly use distance-based neighborhoods.
- Training with k nearest neighbors, or pairs of points.
- Typically Euclidean neighbors
- But in **high d** , your nearest Euclidean neighbor can be **very** different from you...

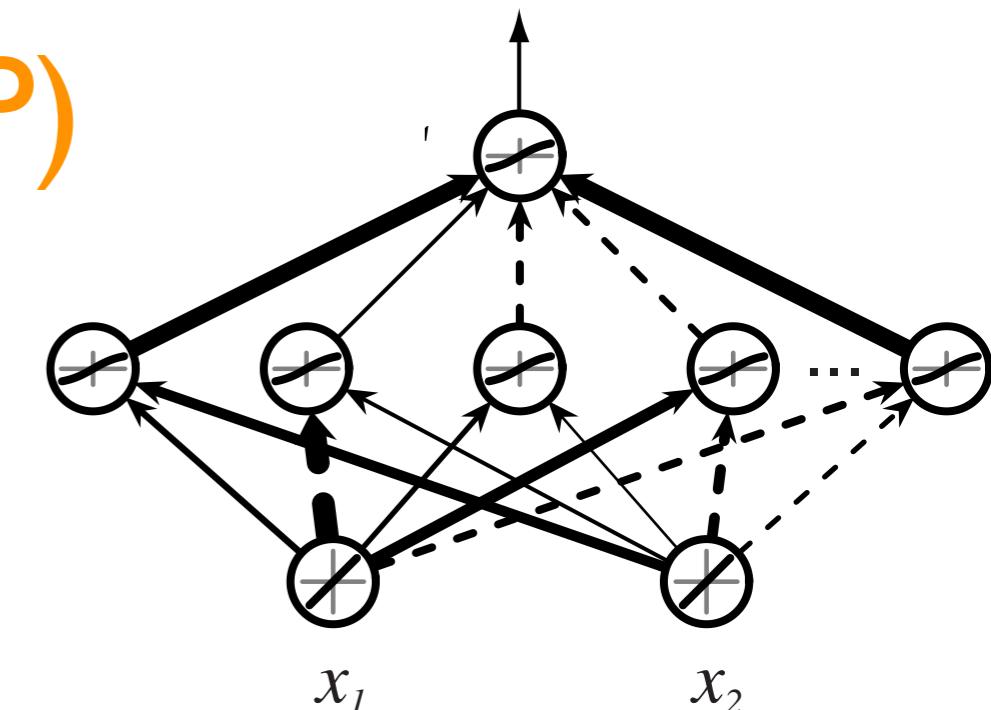


PART II

On Auto-Encoders
and
their regularization,

Multi-Layer Perceptron (MLP)

with **one** hidden layer of size d' neurons



Functional form (parametric):

$$y = f_{\theta}(\mathbf{x}) = \text{sigmoid}(\langle \mathbf{w}, \mathbf{h} \rangle + b)$$

$$\mathbf{h} = \text{sigmoid}(\underbrace{\mathbf{W}^{\text{hidden}} \mathbf{x} + \mathbf{b}^{\text{hidden}}}_{d' \times d \quad d' \times 1})$$

Parameters:

$$\theta = \{\mathbf{W}^{\text{hidden}}, \mathbf{b}^{\text{hidden}}, \mathbf{w}, b\}$$

Optimizing parameters on training set (*training the network*):

$$\theta^* = \arg \min_{\theta} \hat{R}_{\lambda}(f_{\theta}, D_n)$$

$$\mathcal{J}_{\text{MLP}}(\theta) = \left(\sum_{(x,t) \in D} L(t, f_{\theta}(x)) \right) + \lambda \Omega(\theta)$$

empirical risk

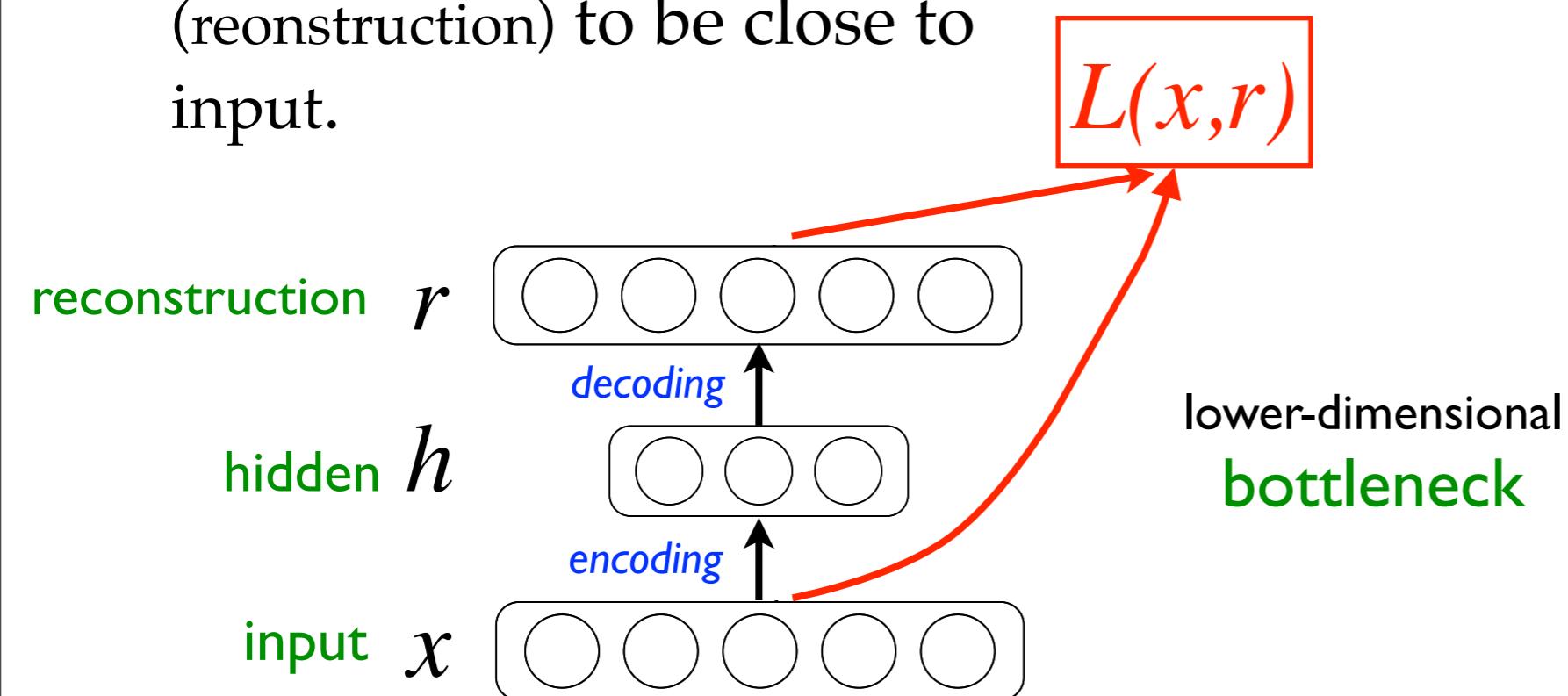
regularization term
(weight decay)

Autoencoders: MLPs used for «unsupervised» representation learning

- Make **output layer same size as input layer**
- Have **target = input**
- **Loss** encourages output (reconstruction) to be close to input.

Autoencoders are also called

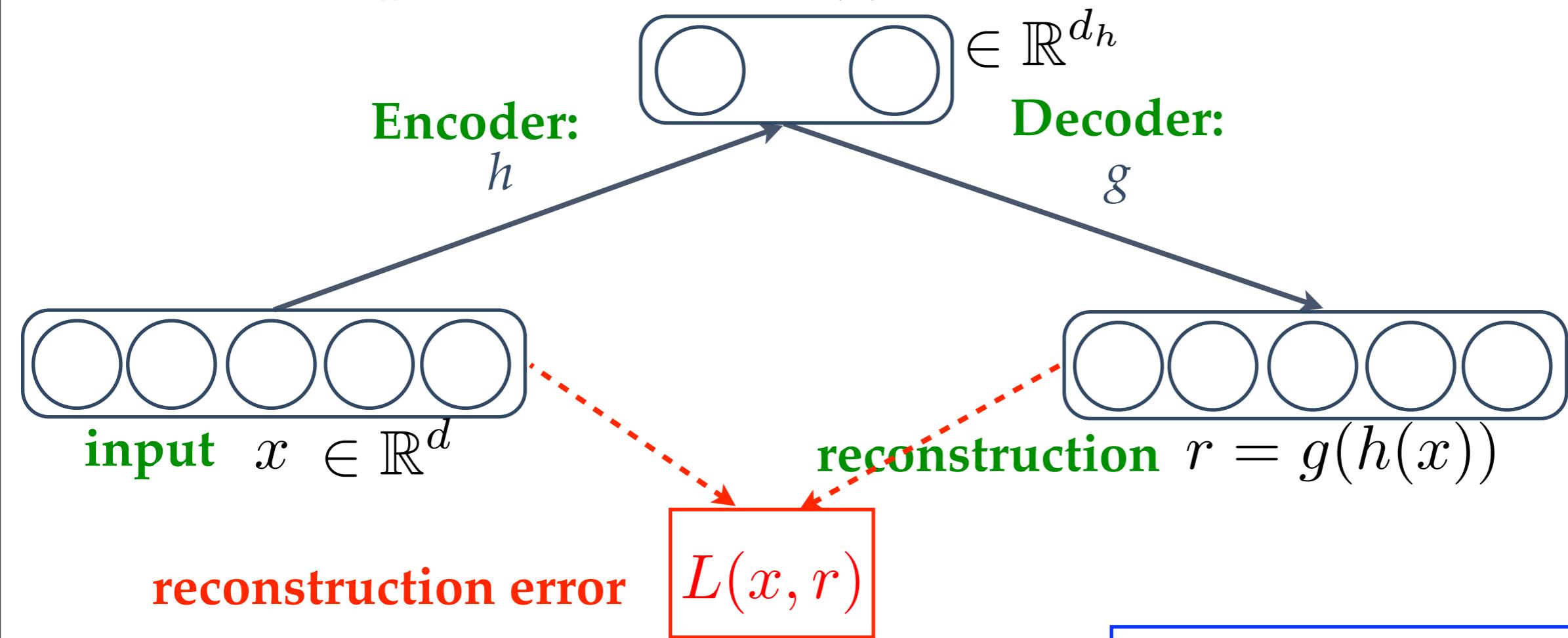
- Autoencoders
- Auto-associators
- Diabolo networks
- Sandglass-shaped net



The Diabolo

Auto-Encoders (AE) for learning representations

hidden representation $\mathbf{h} = h(\mathbf{x})$



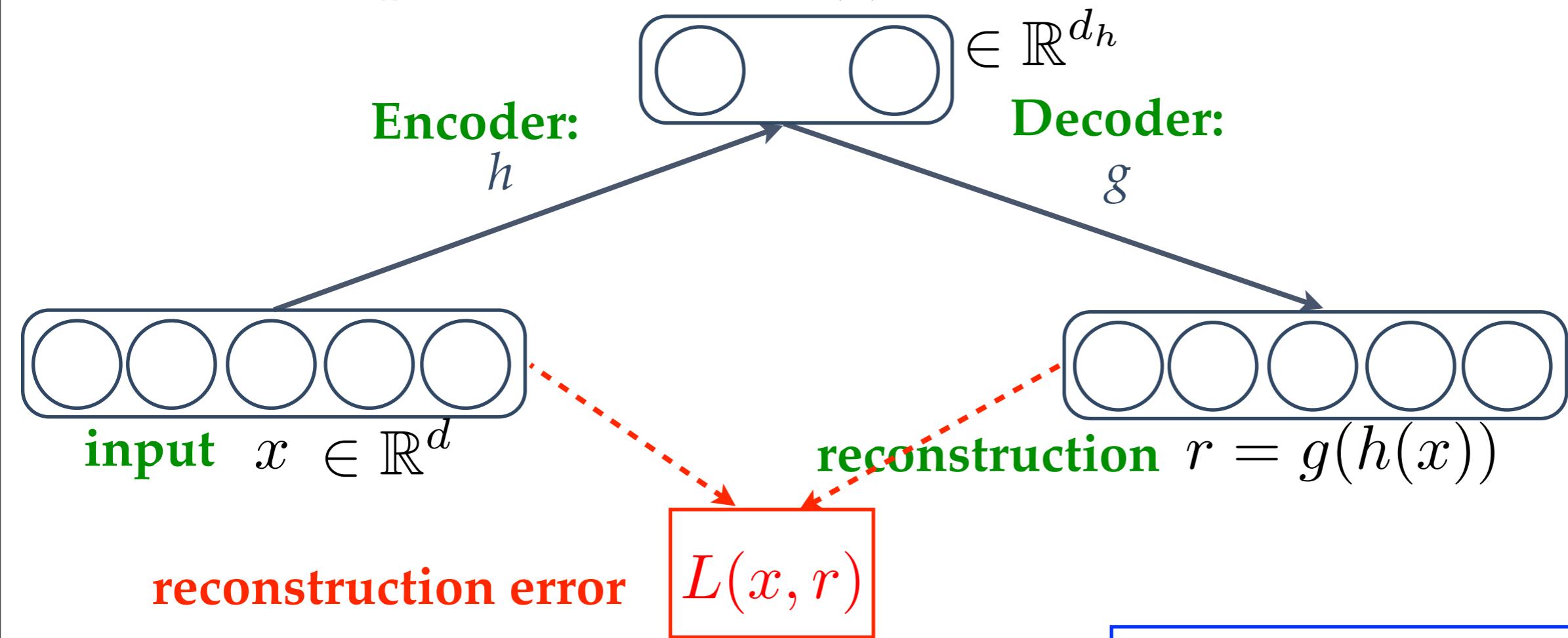
Minimize

$$\mathcal{J}_{\text{AE}} = \sum_{x \in D} L(x, g(h(x)))$$

Auto-Encoders (AE) for learning representations

Typical form

hidden representation $\mathbf{h} = h(\mathbf{x})$



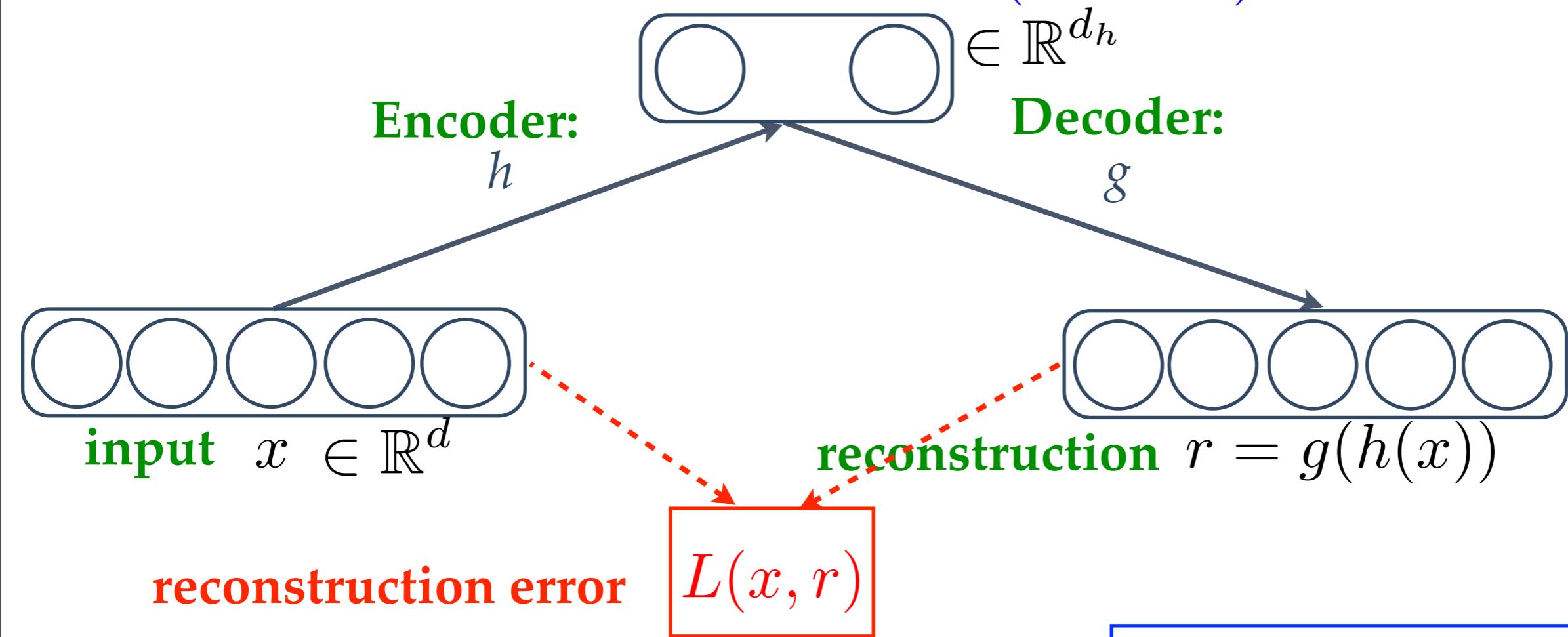
Minimize

$$\mathcal{J}_{\text{AE}} = \sum_{x \in D} L(x, g(h(x)))$$

Auto-Encoders (AE) for learning representations

Typical form

hidden representation $\mathbf{h} = h(\mathbf{x}) = s(W\mathbf{x} + b)$



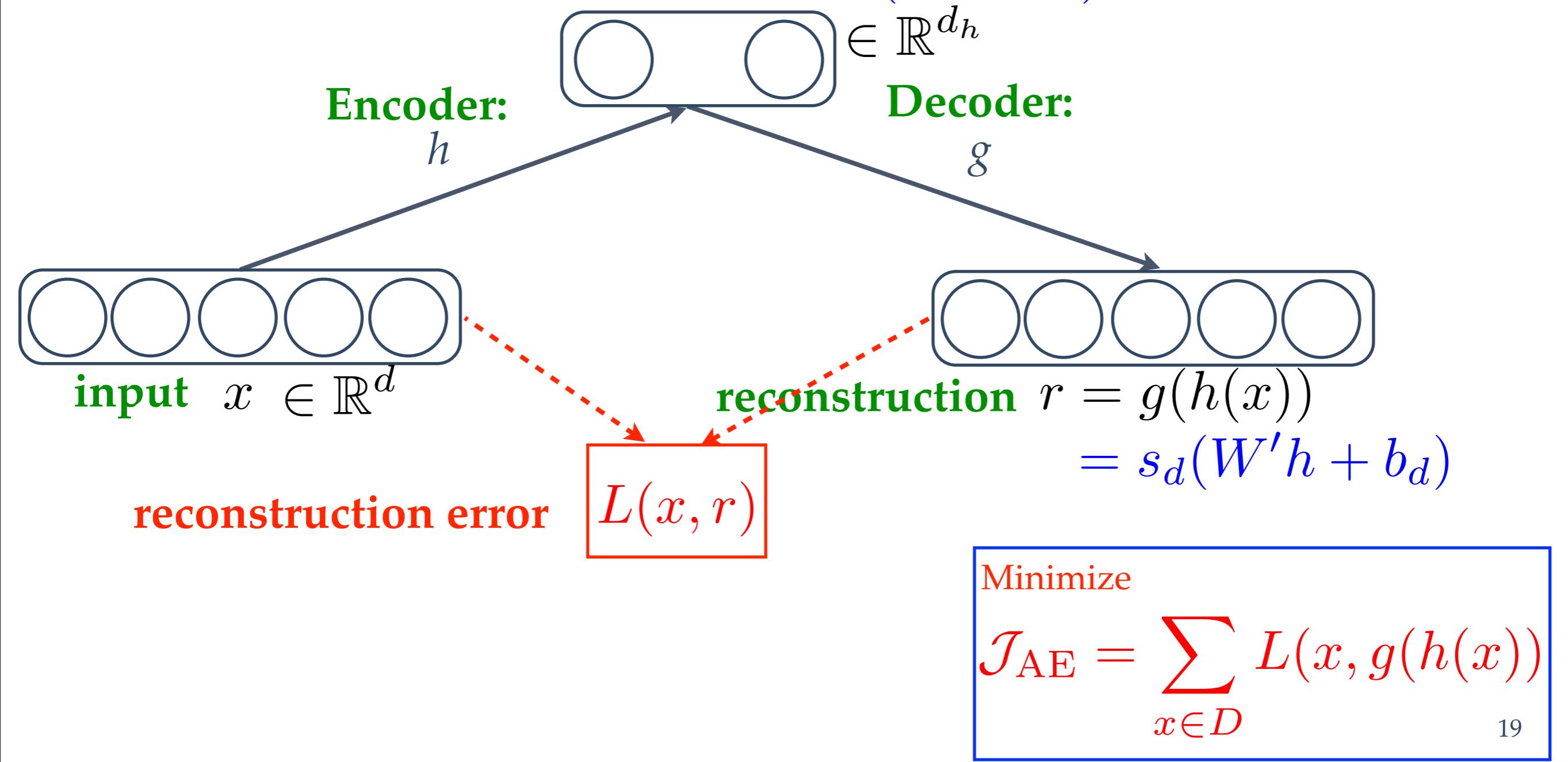
Minimize

$$\mathcal{J}_{\text{AE}} = \sum_{x \in D} L(x, g(h(x)))$$

Auto-Encoders (AE) for learning representations

Typical form

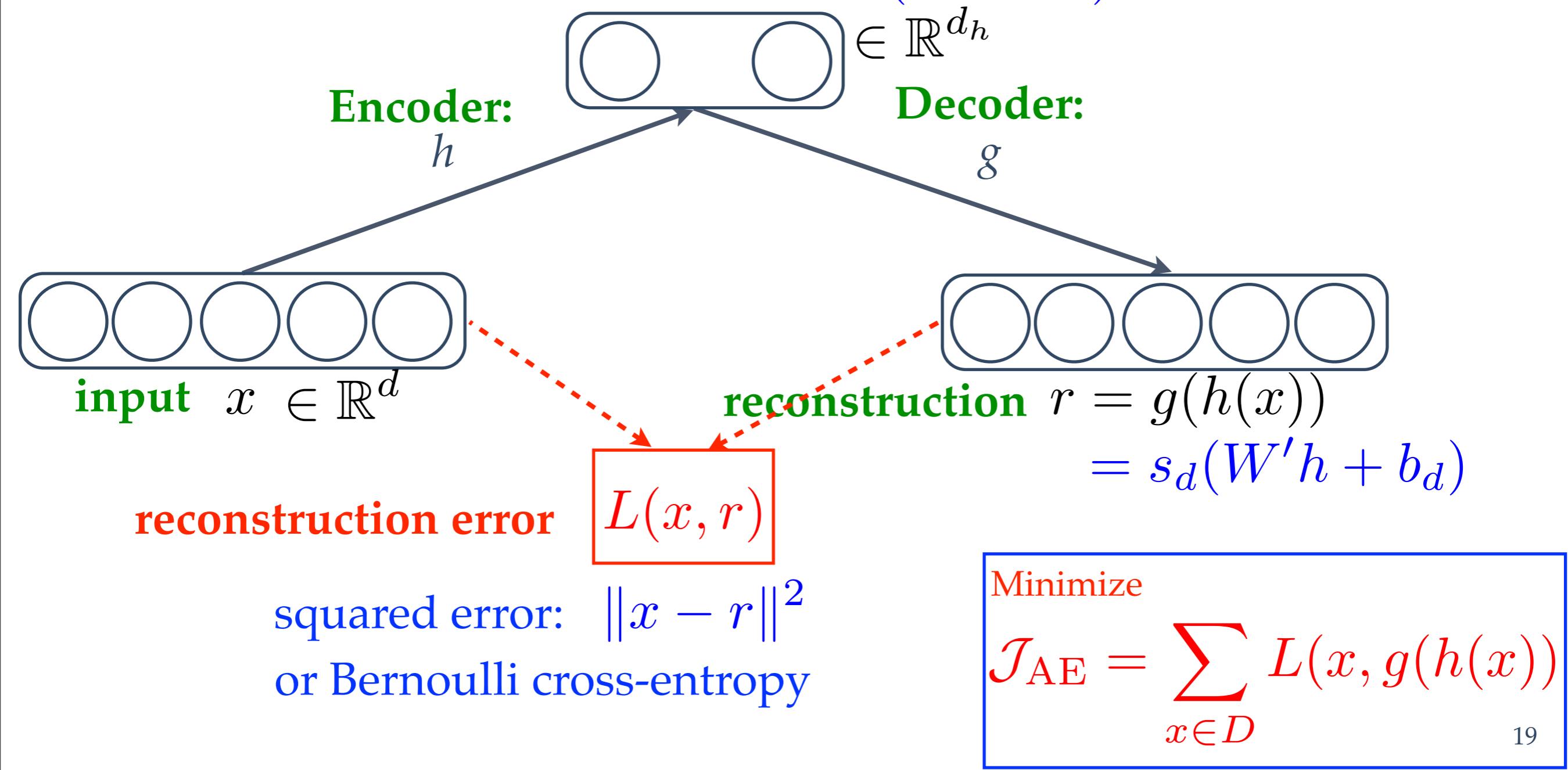
hidden representation $\mathbf{h} = h(\mathbf{x}) = s(W\mathbf{x} + b)$



Auto-Encoders (AE) for learning representations

Typical form

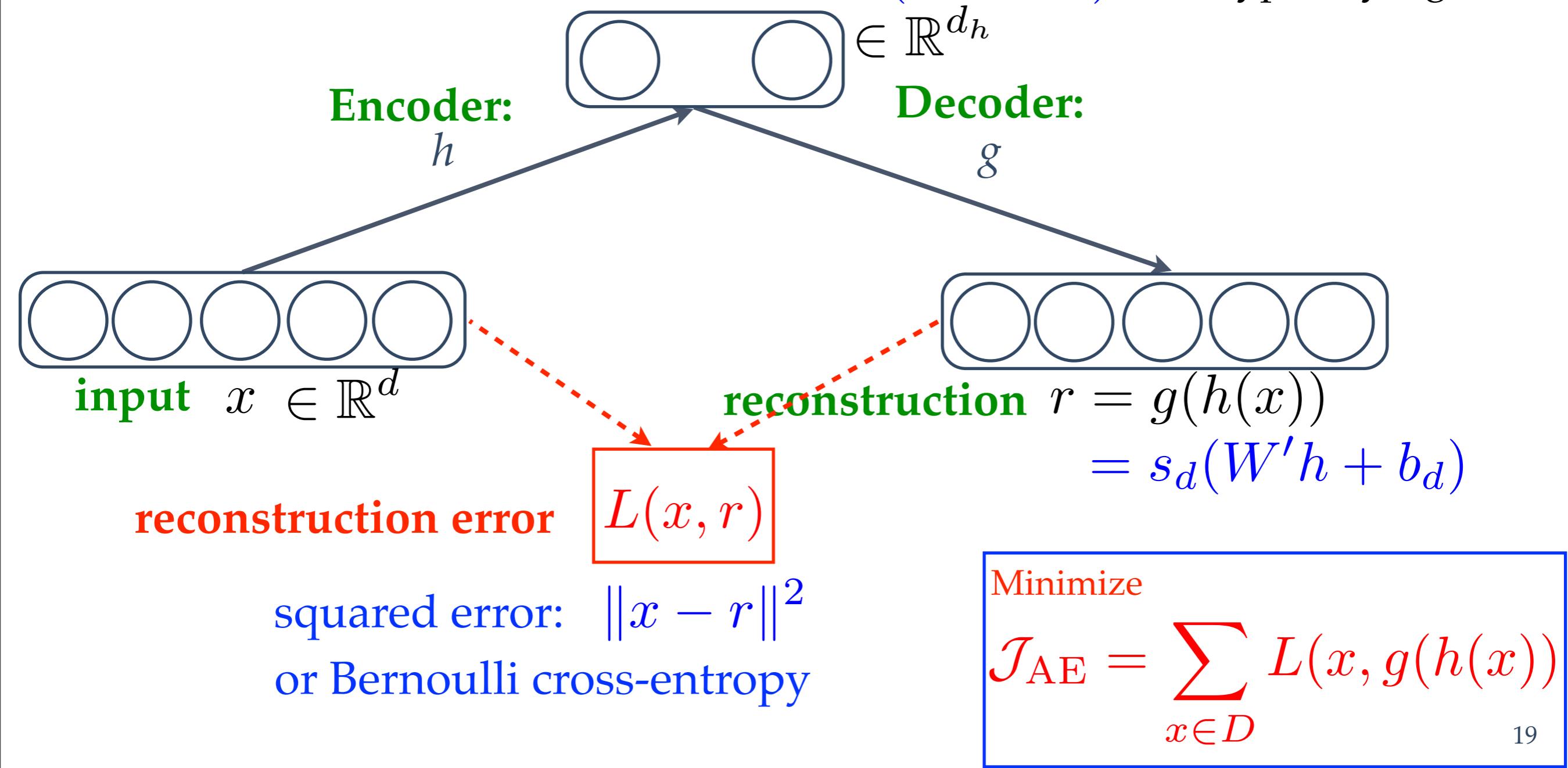
hidden representation $\mathbf{h} = h(\mathbf{x}) = s(W\mathbf{x} + b)$



Auto-Encoders (AE) for learning representations

Typical form

hidden representation $\mathbf{h} = h(\mathbf{x}) = s(W\mathbf{x} + b)$ s is typically sigmoid



conection between

Linear auto-encoders and PCA

$d_h < d$ (bottleneck, undercomplete representation):

- With linear neurons and squared loss
 ➡ autoencoder learns same **subspace** as PCA
- Also true with a single sigmoidal hidden layer,
if using linear output neurons with squared loss
[Baldi & Hornik 89] and untied weights.
- Won't learn the exact same **basis** as PCA,
but W will span the same **subspace**.

similarity between Auto-encoders and RBM

Consider an auto-encoder MLP

- with a single hidden layer with **sigmoid** non-linearity
- and **sigmoid output** non-linearity.
- Tie encoder and decoder weights: $W' = W^T$.

Autoencoder:

$$h_i = s(W_i x + b_i)$$

$$r_j = s(W_j^T h + b_{dj})$$

RBM:

$$P(h_i=1 | v) = s(W_i v + c_i)$$

$$P(v_j=1 | h) = s(W_j^T h + b_j)$$

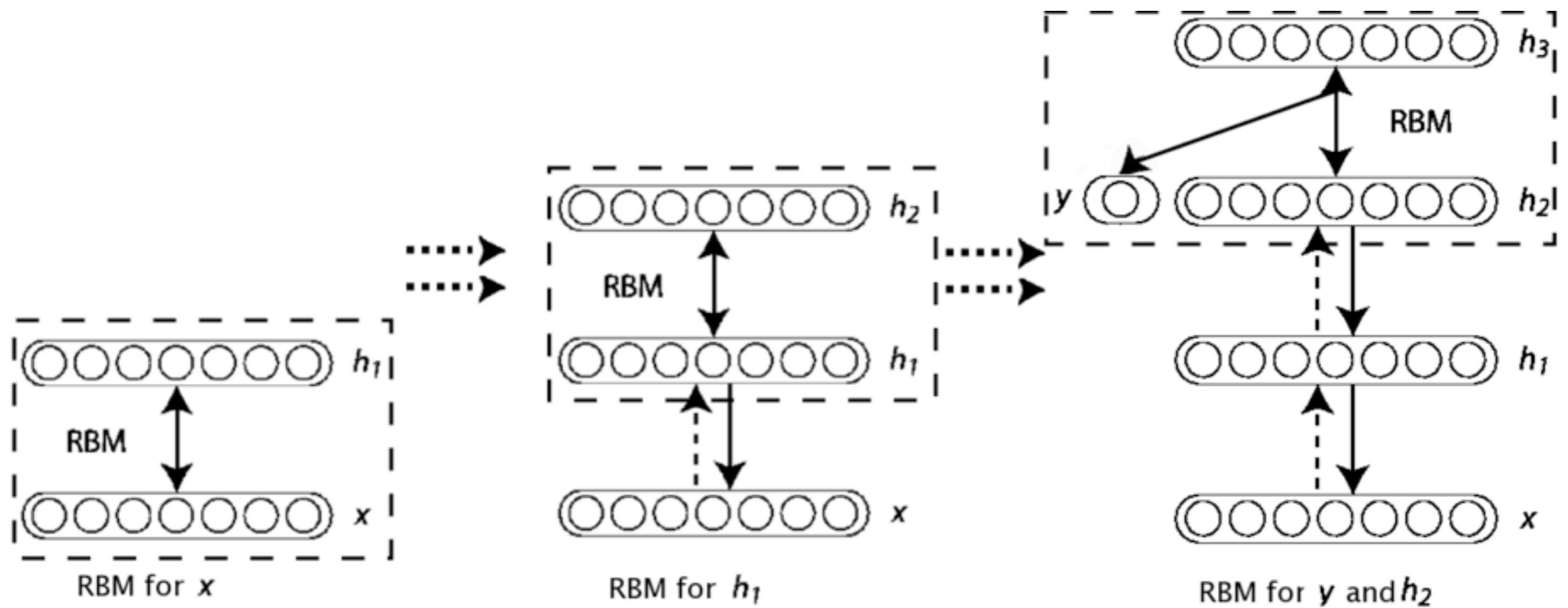
Differences: deterministic mapping
h is a function of x.

stochastic mapping
h is a random variable

Greedy Layer-Wise Pre-training with RBMs

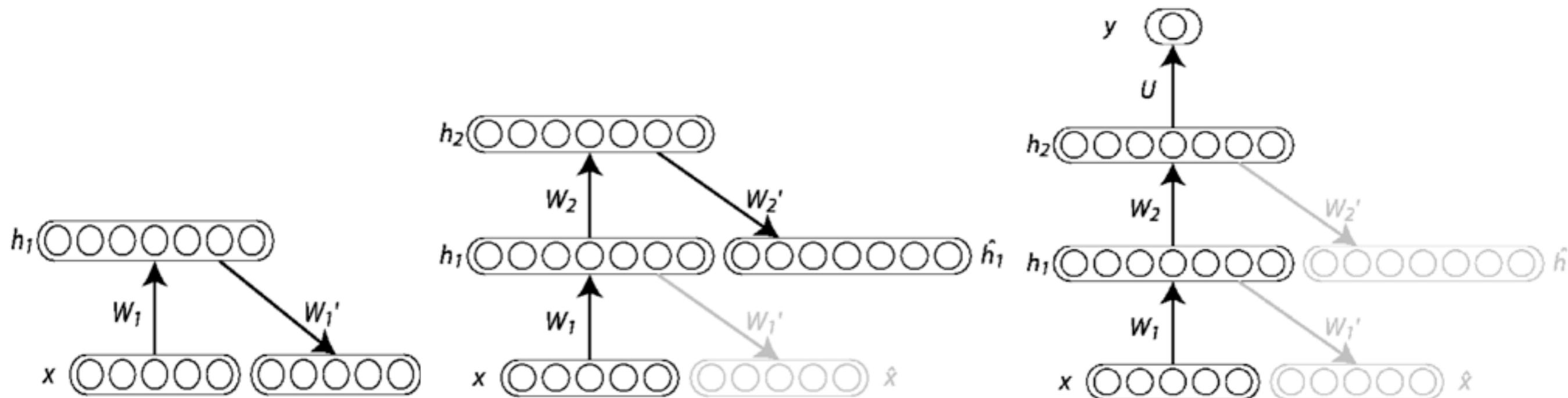
Stacking Restricted Boltzmann Machines (RBM)

⇒ Deep Belief Network (DBN) [Hinton et al. 2006]



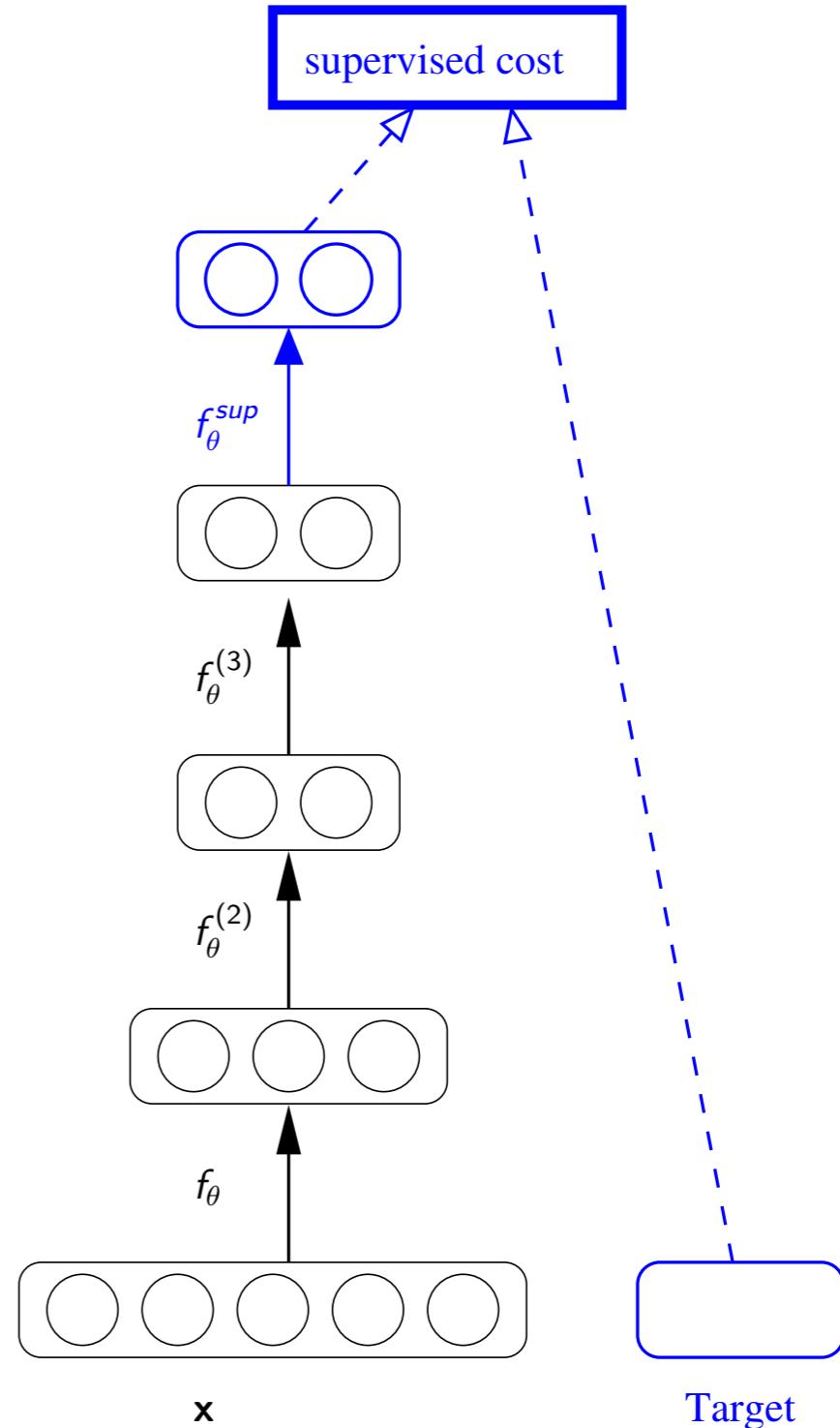
Greedy Layer-Wise Pre-training with Auto-Encoders

Stacking basic Auto-Encoders [Bengio et al. 2007]



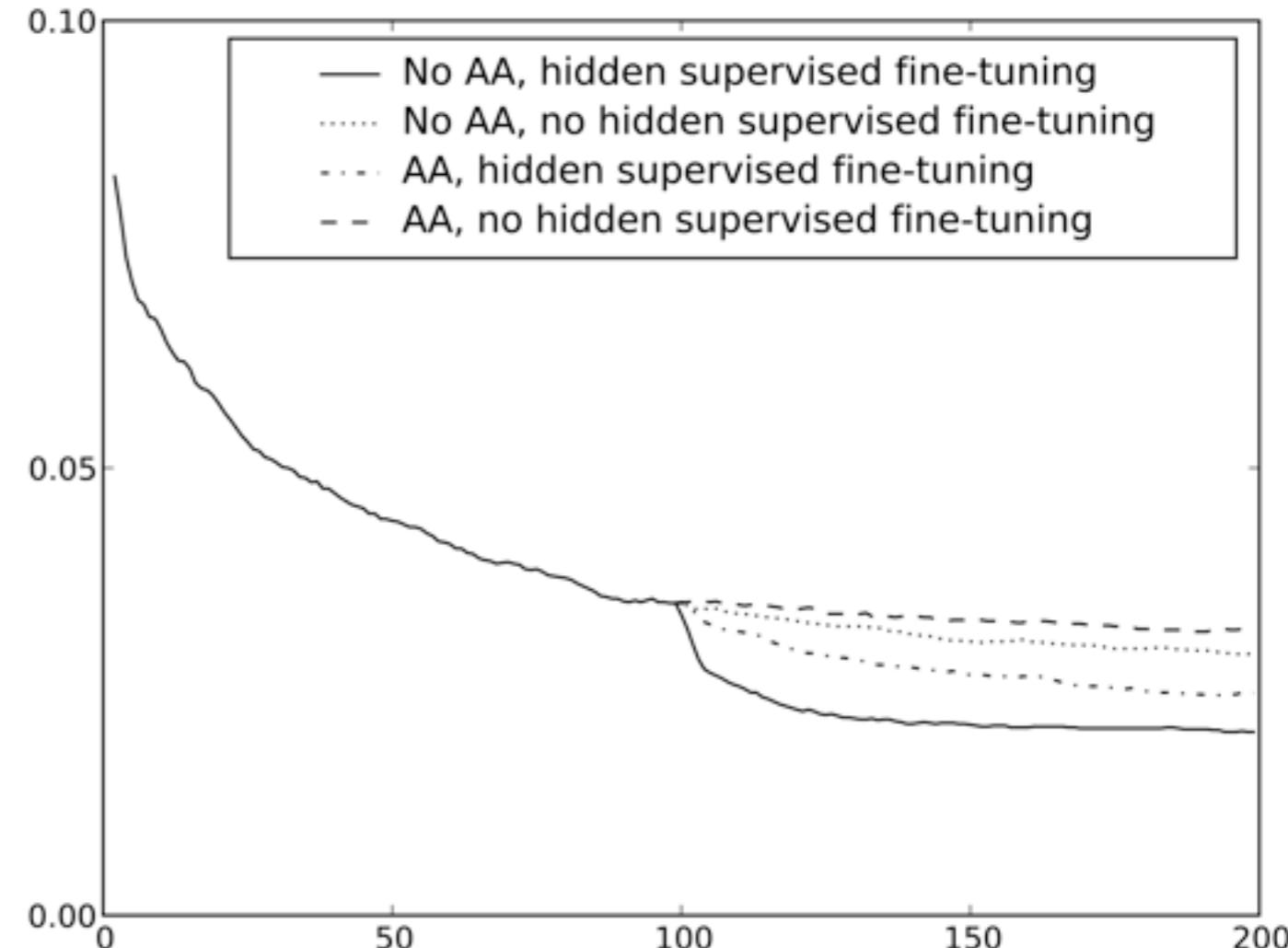
Supervised fine-tuning

- Initial deep mapping was learnt in an **unsupervised** way.
- **initialization** for a **supervised** task.
- Output layer** gets added.
- Global fine tuning by gradient descent on **supervised criterion**.



Supervised Fine-Tuning is Important

- Greedy layer-wise unsupervised pre-training phase with RBMs or auto-encoders on MNIST
- Supervised phase with or without unsupervised updates, with or without fine-tuning of hidden layers



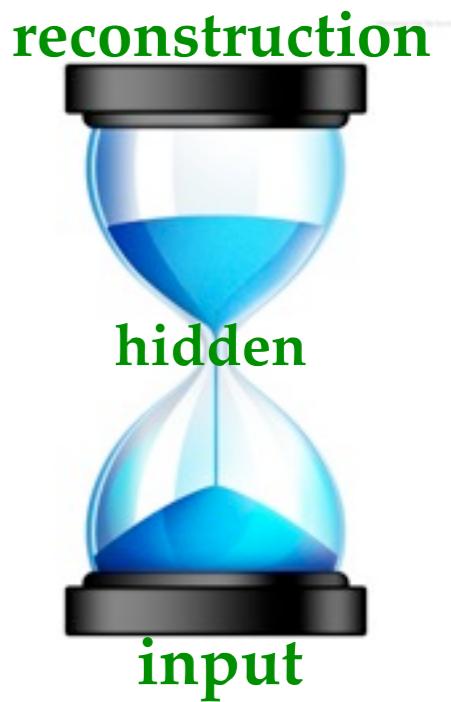
Classification performance on benchmarks:

- Pre-training basic auto-encoder stack **better** than no pre-training
- Basic auto-encoder stack **almost** matched RBM stack...

Basic auto-encoders not as good feature learners as RBMs...

What's the problem?

- ⊕ Traditional autoencoders were for **dimensionality reduction** ($d_h < d_x$)
- ⊕ Deep learning success seems to depend on ability to learn **overcomplete representations** ($d_h > d_x$)
- ⊕ Overcomplete basic autoencoder yields trivial useless solutions: **identity mapping!**
- ⊕ Need for alternative **regularization/ constraining**

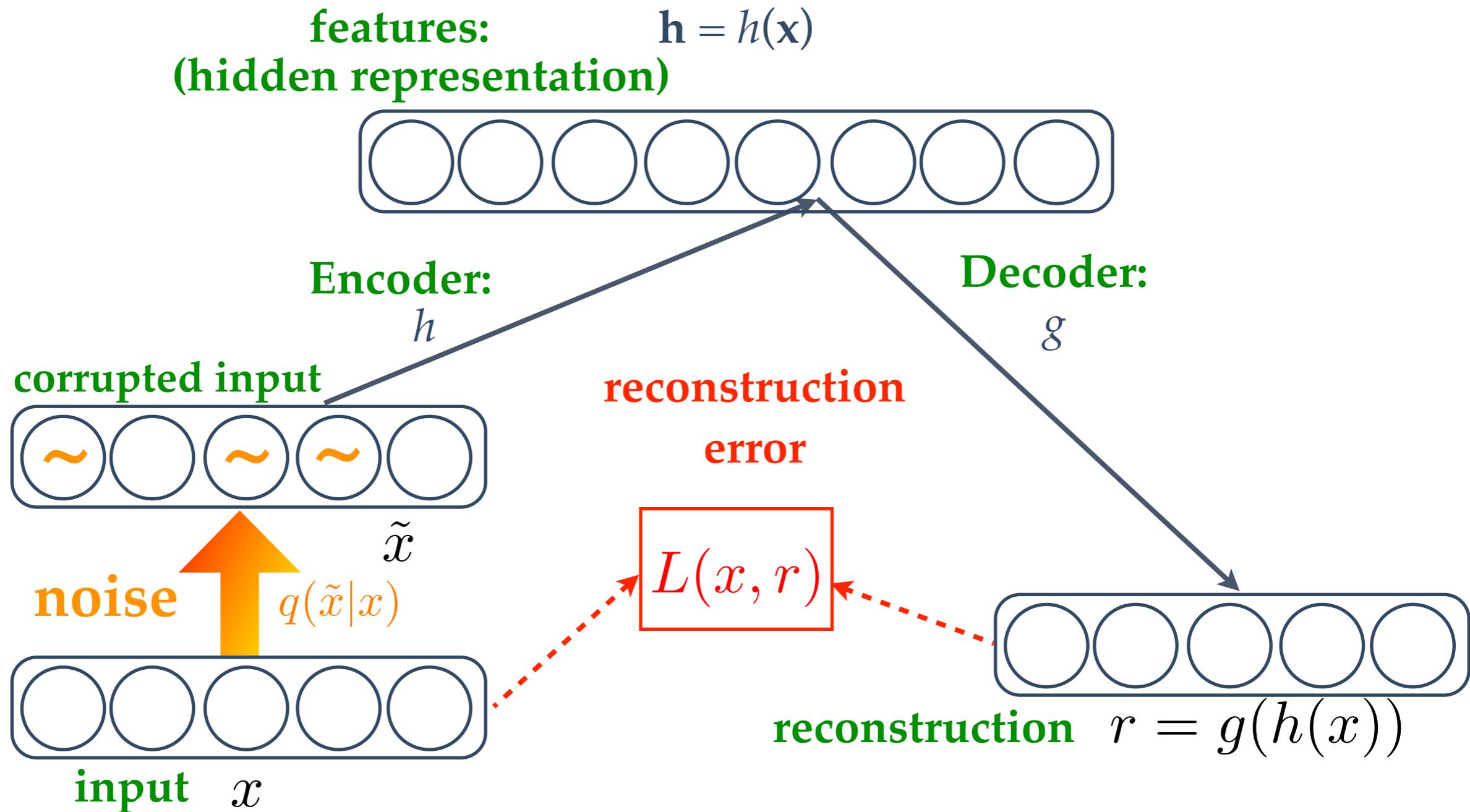


Denoising auto-encoders: motivation

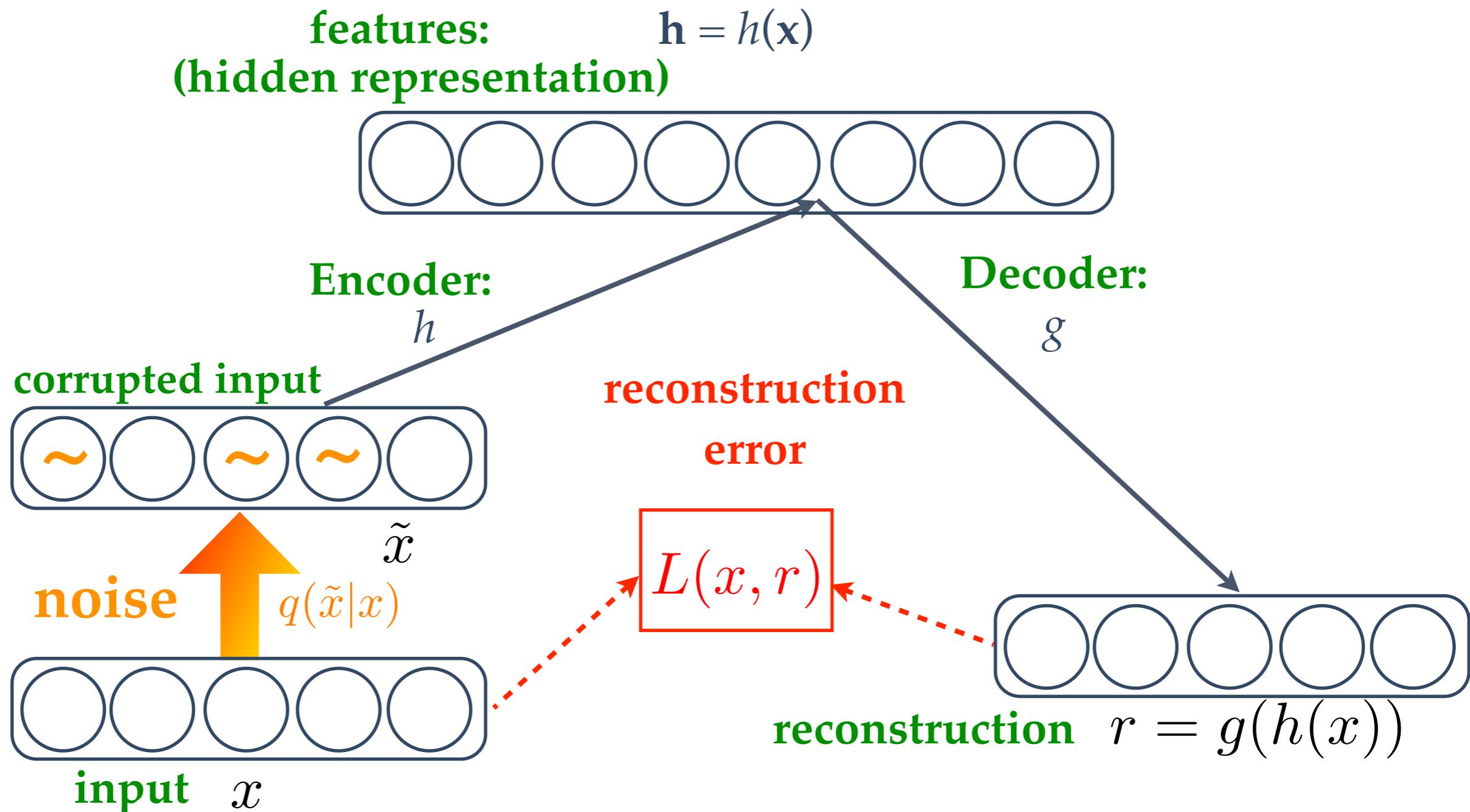
(Vincent, Larochelle, Bengio, Manzagol, ICML 2008)

- ✿ Simple idea «destroying information» of randomly selected input features; train to restore it.
=> 0-masking noise (now called «dropout» noise)
- ✿ Denoising corrupted input is a vastly **more challenging task** than mere reconstruction.
- ✿ Even in widely **over-complete case...**
it **must** learn intelligent encoding/decoding.
- ✿ Will encourage **representation that is robust** to small perturbations of the input.

Denoising auto-encoder (DAE)



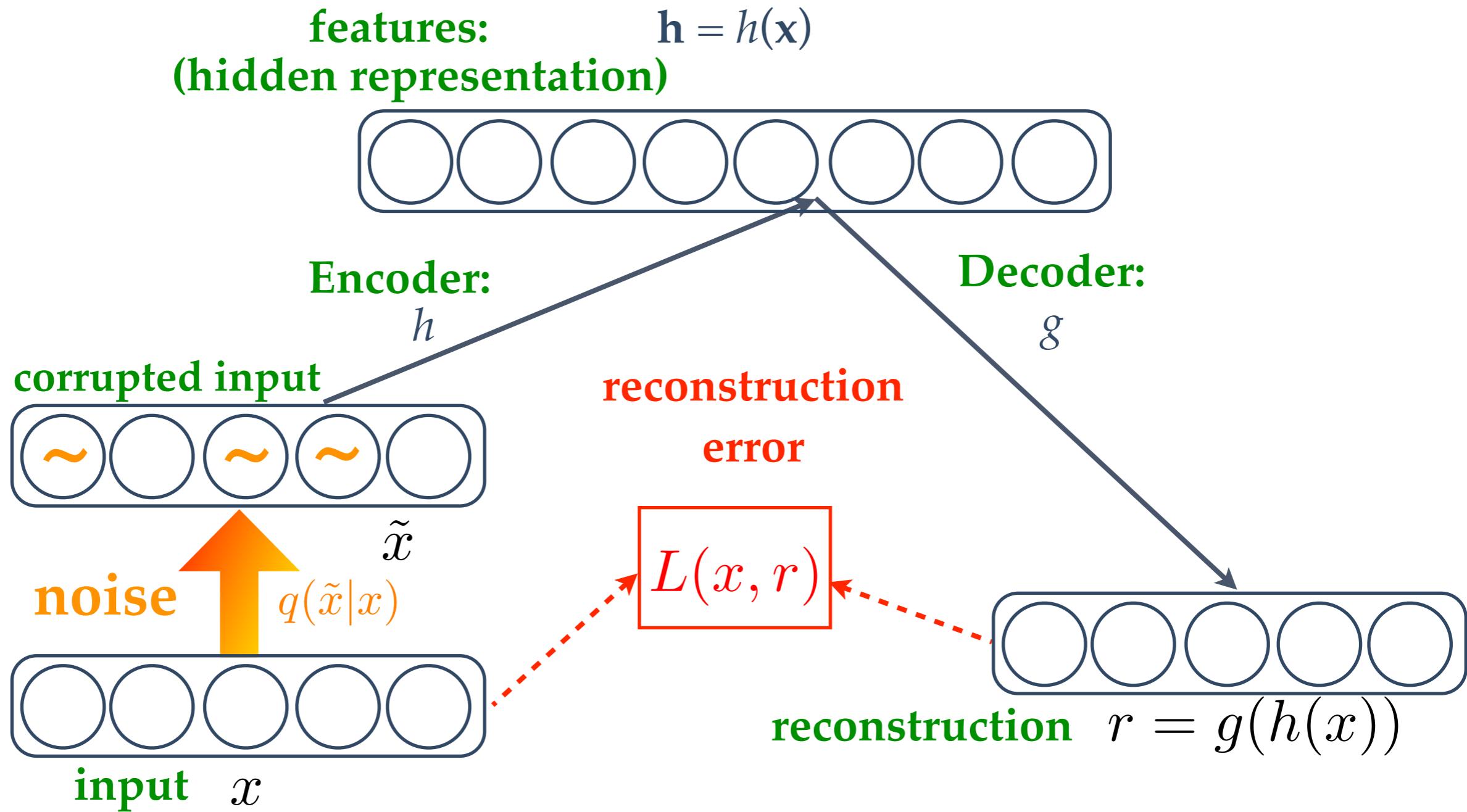
Denoising auto-encoder (DAE)



Minimize:

$$\mathcal{J}_{\text{DAE}}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))]$$

Denoising auto-encoder (DAE)



- learns **robust & useful** features
- easier to train than RBM features
- yield **similar or better classification** performance (as deep net pre-training)

Minimize:

$$\mathcal{J}_{\text{DAE}}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))]$$

Denoising auto-encoder (DAE)

- Autoencoder training minimizes:

$$\mathcal{J}_{\text{AE}}(\theta) = \sum_{x \in D} L(x, g(h(x)))$$

- Denoising autoencoder training minimizes

$$\mathcal{J}_{\text{DAE}}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))]$$

Cannot compute expectation exactly

⇒ use stochastic gradient descent,
sampling corrupted inputs $\tilde{x}|x$

Denoising auto-encoder (DAE)

- Autoencoder training minimizes:

$$\mathcal{J}_{\text{AE}}(\theta) = \sum_{x \in D} L(x, g(h(x)))$$

- Denoising autoencoder training minimizes

$$\mathcal{J}_{\text{DAE}}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))]$$

Cannot compute expectation exactly

⇒ use stochastic gradient descent,
sampling corrupted inputs $\tilde{x}|x$

Possible corruptions q:

- zeroing pixels at random
(now called «dropout» noise)
- additive Gaussian noise
- salt-and-pepper noise
- ...

Denoising auto-encoder (DAE)

- Autoencoder training minimizes:

$$\mathcal{J}_{AE}(\theta) = \sum_{x \in D} L(x, g(h(\tilde{x}))$$

- Denoising autoencoder training minimizes

$$\mathcal{J}_{DAE}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))]$$

Cannot compute expectation exactly

⇒ use stochastic gradient descent,

sampling corrupted inputs $\tilde{x}|x$

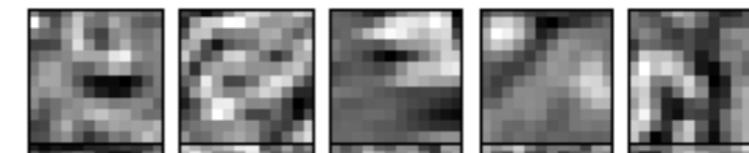
Possible corruptions q:

- zeroing pixels at random
(now called «dropout» noise)
- additive Gaussian noise
- salt-and-pepper noise
- ...

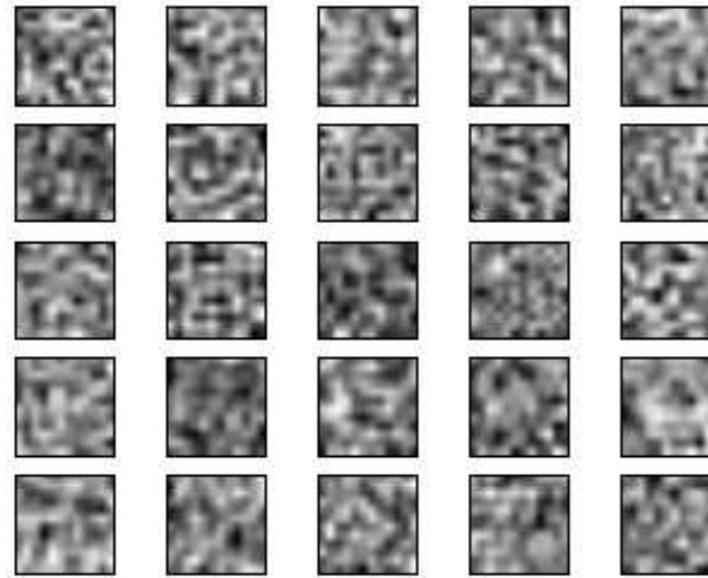
Learned filters

a) Natural image patches

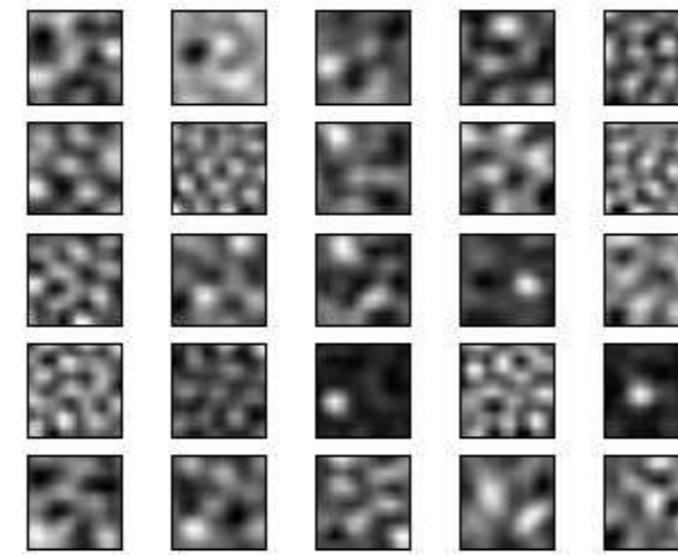
e.g.:



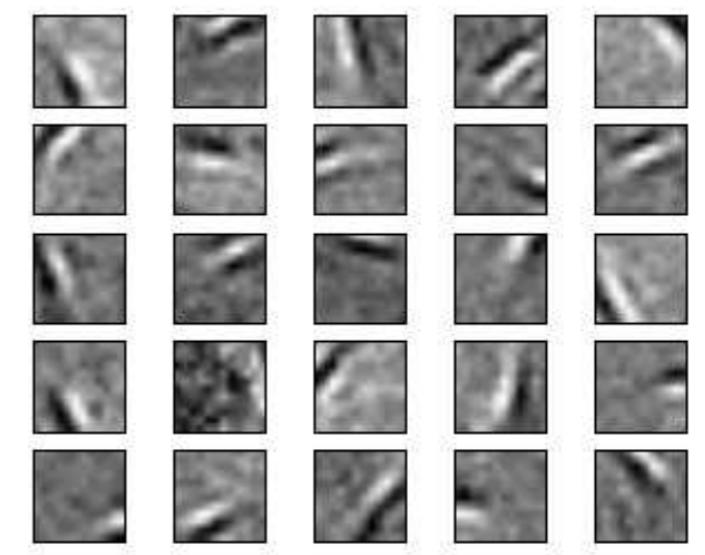
AE



AE with weight decay



DAE

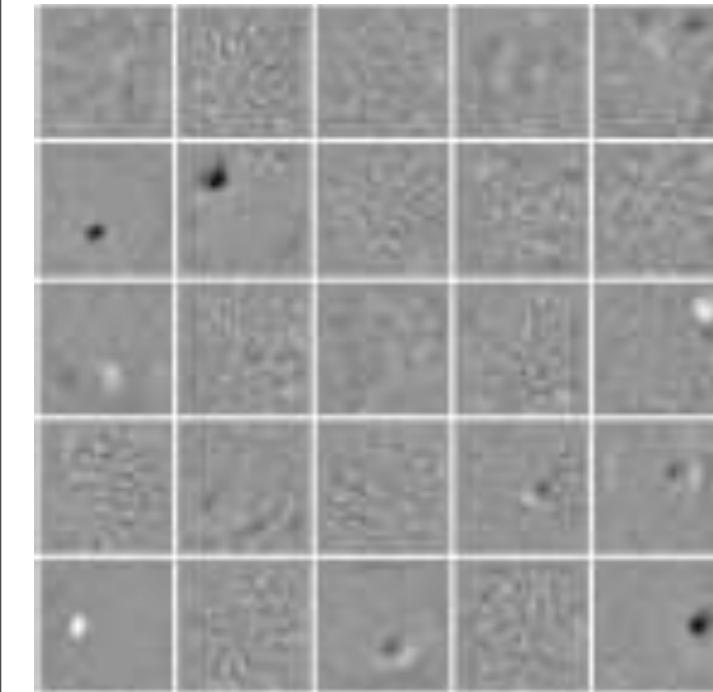


Learned filters

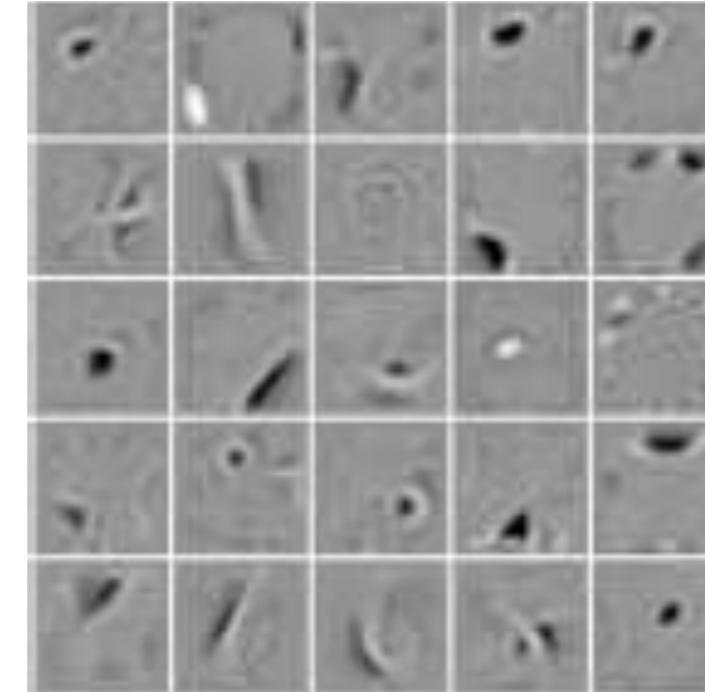
b) MNIST digits

e.g.: 4 3 8 7

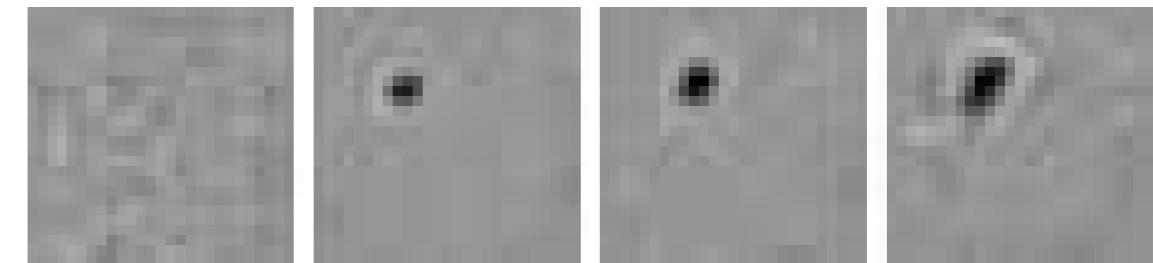
AE



DAE



Increasing noise



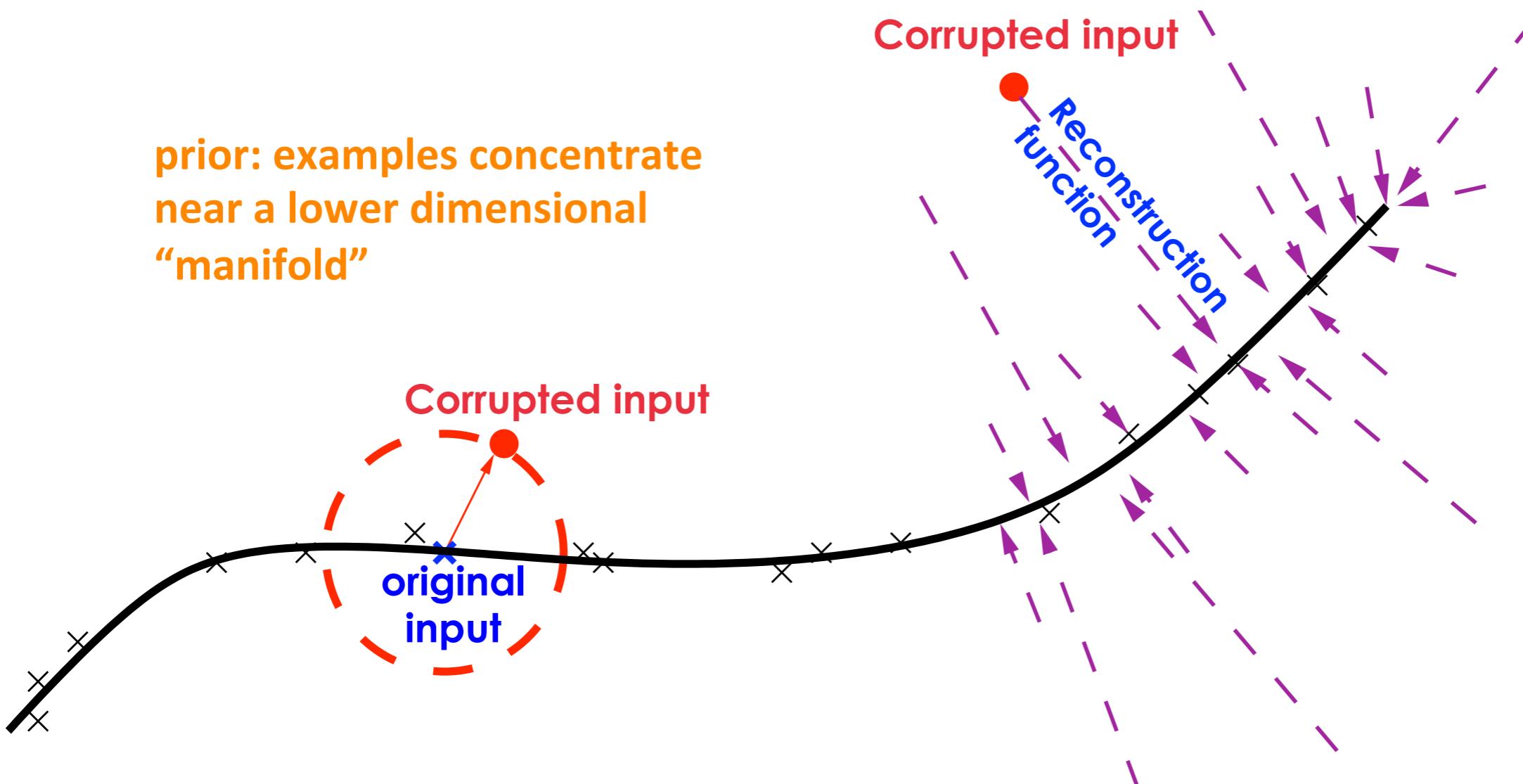
(d) Neuron A (0%, 10%, 20%, 50% corruption)



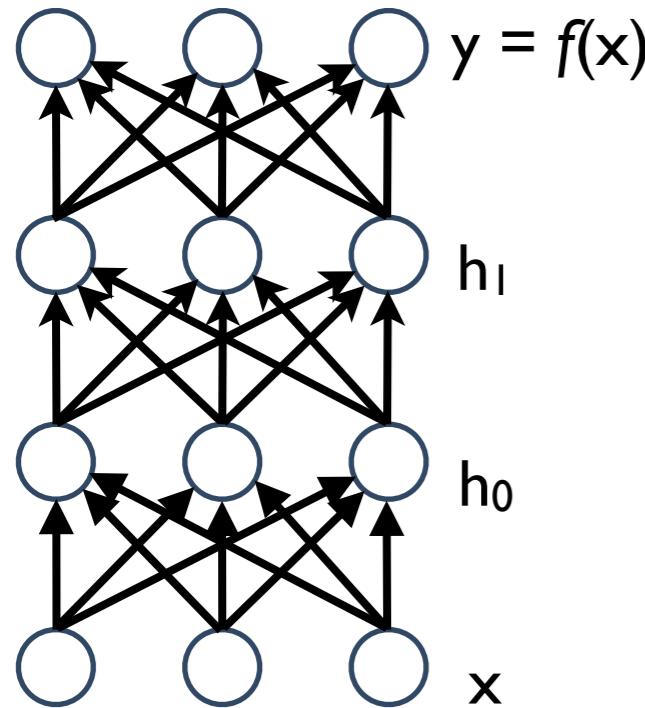
(e) Neuron B (0%, 10%, 20%, 50% corruption)

Denoising auto-encoders: manifold interpretation

- DAE learns to «project back» corrupted input onto manifold.
- Representation $h \approx$ location on the manifold

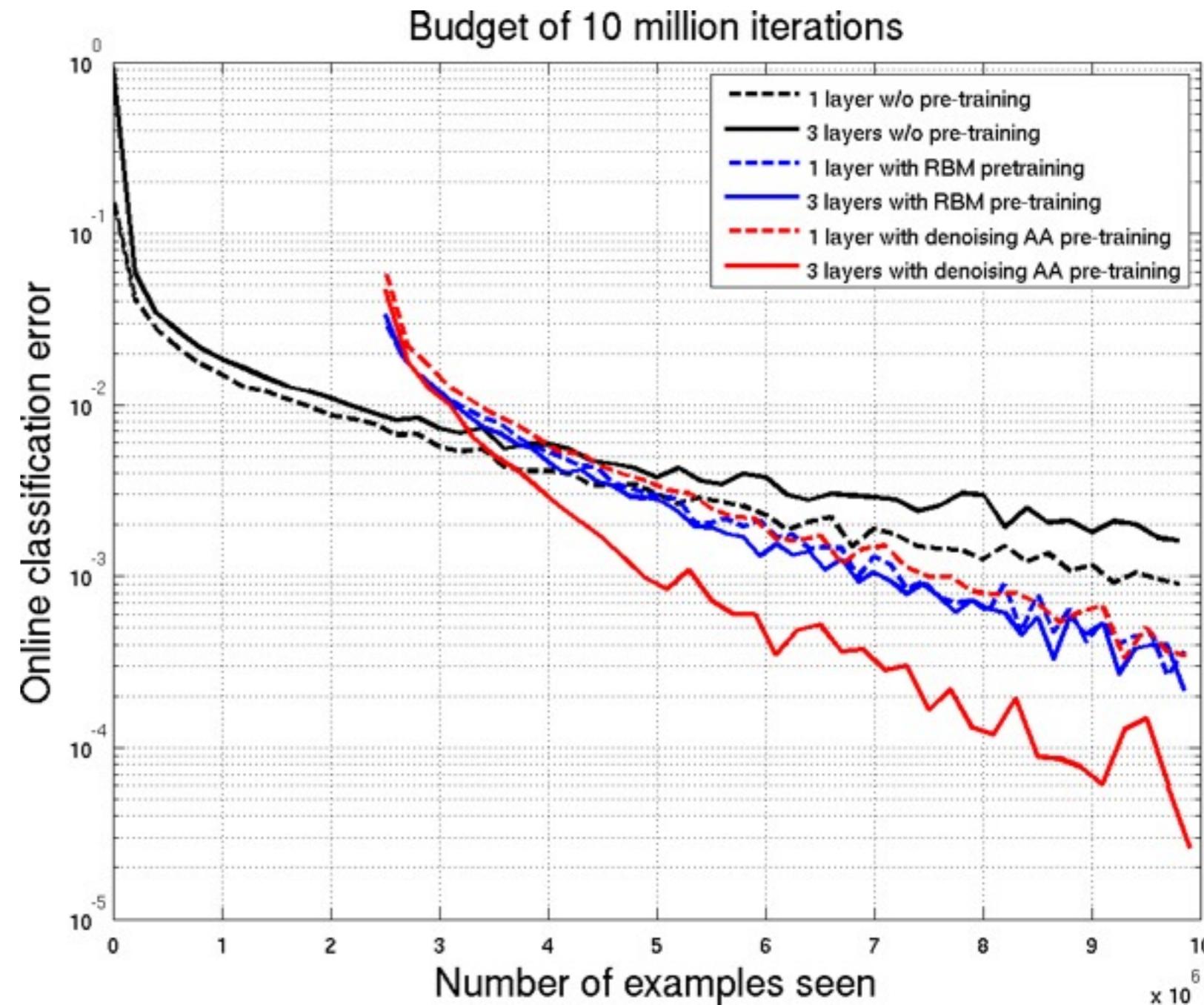


Stacked Denoising Auto-Encoders (SDAE)



Advantages over stacking RBMs

- No partition function, can measure training criterion
- Very flexible: encoder & decoder can use any parametrization (more layers...)
- Performs as well or better than stacking RBMs for unsupervised pre-training



Infinite MNIST

Encouraging representation to be insensitive to corruption

- DAE encourages **reconstruction** to be insensitive to input corruption
- Alternative: encourage **representation** to be **insensitive**

$$\mathcal{J}_{\text{SCAE}}(\theta) = \sum_{x \in D} L(x, g(h(x))) + \lambda \mathbb{E}_{q(\tilde{x}|x)} [\|h(x) - h(\tilde{x})\|^2]$$

Reconstruction error **stochastic regularization term**

- Tied weights i.e. $W' = W^T$ prevent W from collapsing h to 0.

Encouraging representation to be insensitive to corruption

- DAE encourages **reconstruction** to be insensitive to input corruption
- Alternative: encourage **representation** to be insensitive to corruption

$$\mathcal{J}_{\text{SCAE}}(\theta) = \sum_{x \in D} E(x, g(h(x))) + \lambda \mathbb{E}_{q(\tilde{x}|x)} [\|h(x) - h(\tilde{x})\|^2]$$

Reconstruction error stochastic regularization term

regularization

- Tied weights i.e. $W = W^T$ prevent W from collapsing h to 0.

From stochastic to analytic penalty

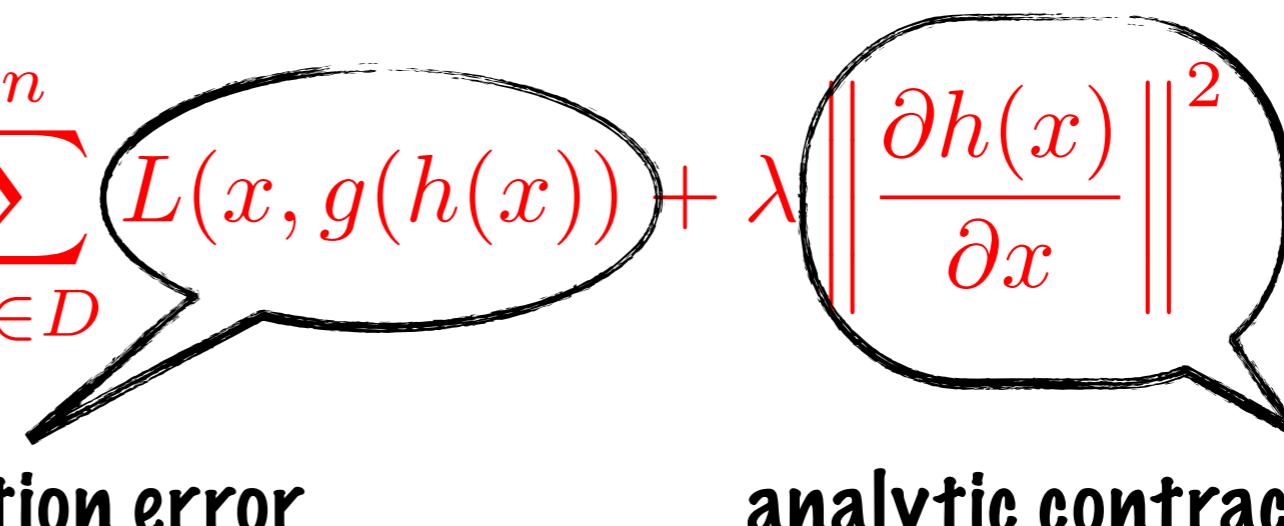
- * SCAE stochastic regularization term: $\mathbb{E}_{q(\tilde{x}|x)} [\|h(x) - h(\tilde{x})\|^2]$
- * For small additive noise $\tilde{x}|x = x + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I)$
- * Taylor series expansion yields $h(x + \epsilon) = h(x) + \frac{\partial h}{\partial x} \epsilon + \dots$
- * It can be showed that

$$\mathbb{E}_{q(\tilde{x}|x)} [\|h(x) - h(\tilde{x})\|^2] \approx \sigma^2 \left\| \frac{\partial h}{\partial x}(x) \right\|_F^2$$


stochastic
(SCAE) analytic
(CAE)

Contractive Auto-Encoder (CAE)

(Rifai, Vincent, Muller, Glorot, Bengio, ICML 2011)

-
- * Minimize $\mathcal{J}_{\text{CAE}} = \sum_{x \in D}^n L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2$


Reconstruction error analytic contractive term
 - * For training examples, encourages both:
 - small reconstruction error
 - representation insensitive to small variations around example

Contractive Auto-Encoder (CAE)

(Rifai, Vincent, Muller, Glorot, Bengio, ICML 2011)

- Minimize $\mathcal{J}_{\text{CAE}} = \sum_{a \in D}^n L(x, g_h(x)) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2$
 - Reconstruction error
 - Analytic contractive term
- For training examples, encourages both:
 - small reconstruction error
 - representation insensitive to small variations around example

~~Analytic regularization~~

Computational considerations CAE for a simple encoder layer

We defined $\mathbf{h} = h(\mathbf{x}) = s(\underbrace{W\mathbf{x} + b}_a)$

Further suppose: s is an elementwise non-linearity
 s' its first derivative.

Let $J(x) = \frac{\partial h}{\partial x}(x)$

$$J_j = s'(b + x^T W_j) W_j \quad \text{where } J_j \text{ and } W_j \text{ represent j}^{\text{th}} \text{ row}$$

CAE penalty is: $\|J\|_F^2 = \sum_{j=1}^{d_h} s'(a_j)^2 \|W_j\|^2$

Same complexity:
 $O(d_h d)$

Compare to L2 weight decay: $\|W\|_F^2 = \sum_{j=1}^{d_h} \|W_j\|^2$

Gradient backprop
wrt parameters:
 $O(d_h d)$

Higher order Contractive Auto-Encoder (CAE+H)

(Rifai, Mesnil, Vincent, Muller, Bengio, Dauphin, Glorot; ECML 2011)

- CAE penalizes Jacobian norm
- We could also penalize higher order derivatives
- Computationally too expensive: second derivative is a 3-tensor, ...
- **Stochastic approach for efficiency:**

Encourage Jacobian at x and at $x+\epsilon$ to be the same.

$$\begin{aligned} \mathcal{J}_{\text{CAE+H}} &= \sum_{x \in D}^n L(x, g(h(x))) + \lambda \left\| \frac{\partial h}{\partial x}(x) \right\|^2 \\ &\quad + \gamma \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2)} \left[\left\| \frac{\partial h}{\partial x}(x) - \frac{\partial h}{\partial x}(x + \epsilon) \right\|^2 \right] \end{aligned}$$

Higher order Contractive Auto-Encoder (CAE+H)

(Rifai, Mesnil, Vincent, Muller, Bengio, Dauphin, Glorot; ECML 2011)

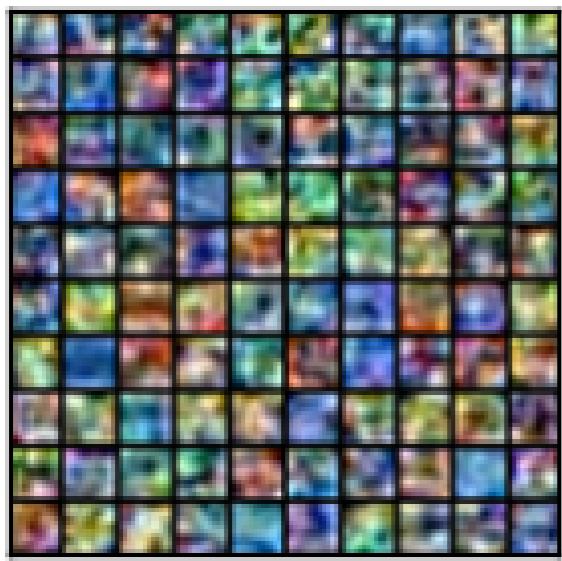
- CAE penalizes Jacobian norm
- We could also penalize higher order derivatives
- Computationally too expensive: second derivative is a 3-tensor, ...
- **Stochastic approach for efficiency:**
Encourage Jacobian at x and at $x+\epsilon$ to be the same.

$$\mathcal{J}_{\text{CAE+H}} = \sum_{x \in D} L(x, g(h(x))) + \lambda \left\| \frac{\partial h}{\partial x}(x) \right\|^2$$

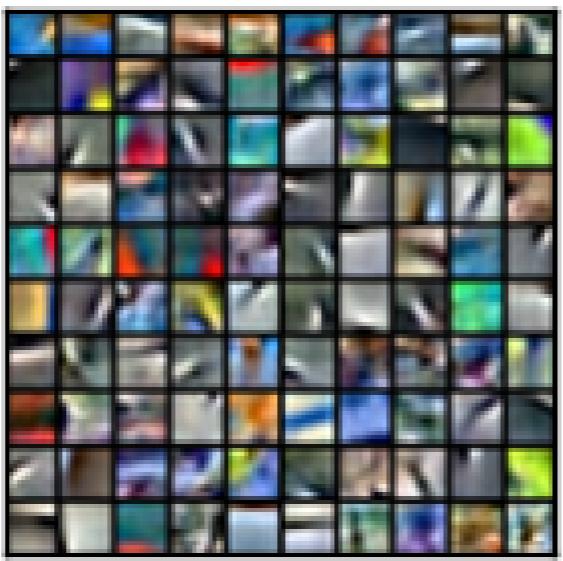
$$+ \gamma \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2)} \left[\left\| \frac{\partial h}{\partial x}(x) - \frac{\partial h}{\partial x}(x + \epsilon) \right\|^2 \right]$$

Learned filters

CIFAR-10



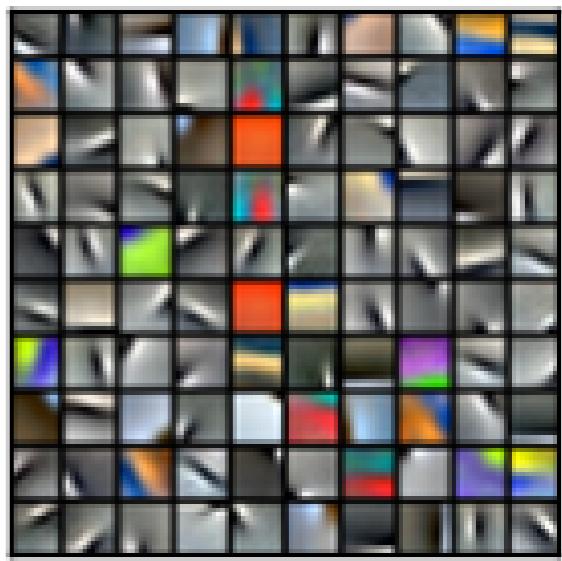
AE



DAE

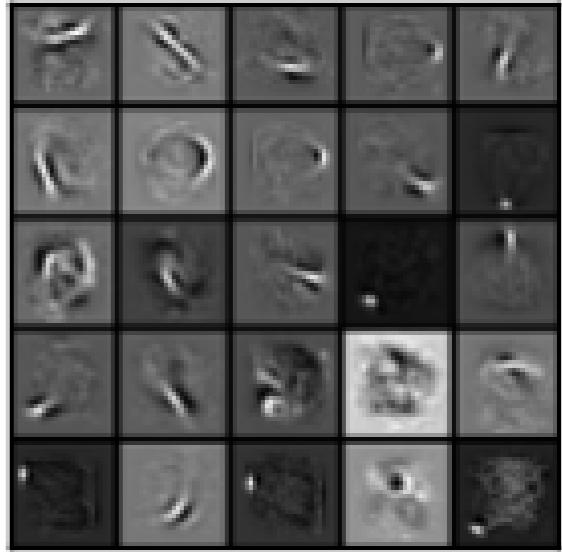
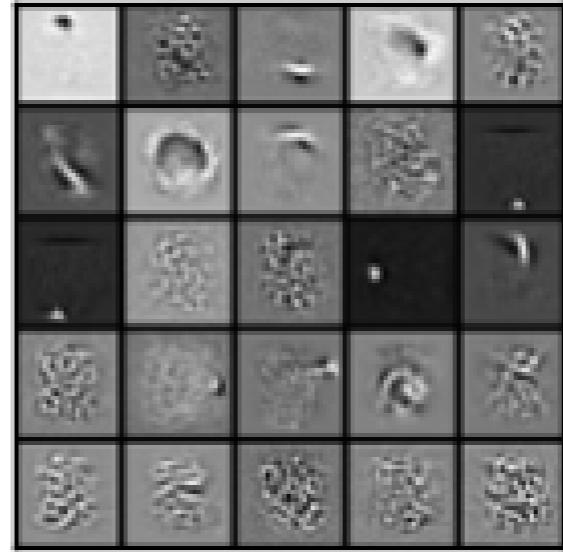
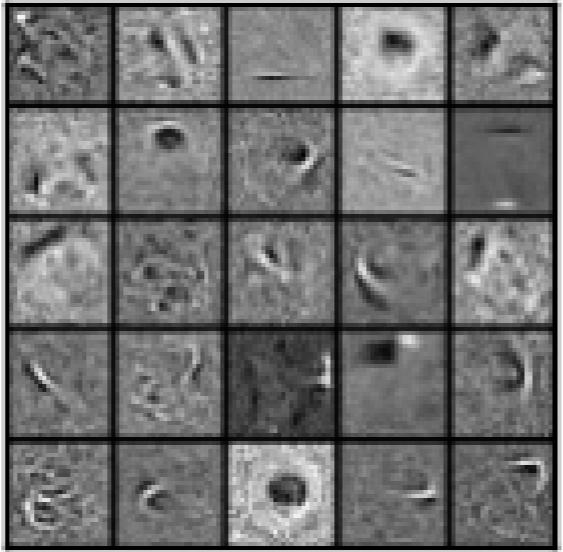
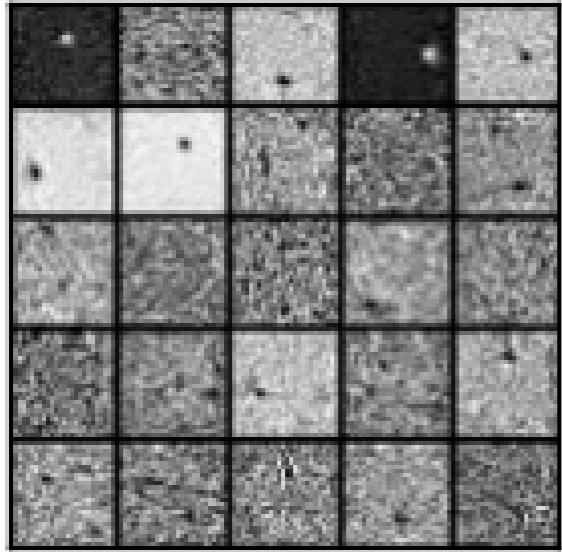


CAE



CAE+H

MNIST



CAE must capture manifold directions

$$\mathcal{J}_{\text{CAE}} = \sum_{x \in D}^n L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2$$

Reconstruction

\Leftarrow tradeoff \Rightarrow

(warning: may require tied weights)

Contraction

pressure to be insensitive
to *all* directions



CAE must capture manifold directions

$$\mathcal{J}_{\text{CAE}} = \sum_{x \in D}^n L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2$$

Reconstruction

\Leftarrow tradeoff \Rightarrow

(warning: may require tied weights)

Contraction

pressure to be insensitive
to *all* directions



CAE must capture manifold directions

$$\mathcal{J}_{\text{CAE}} = \sum_{x \in D}^n L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2$$

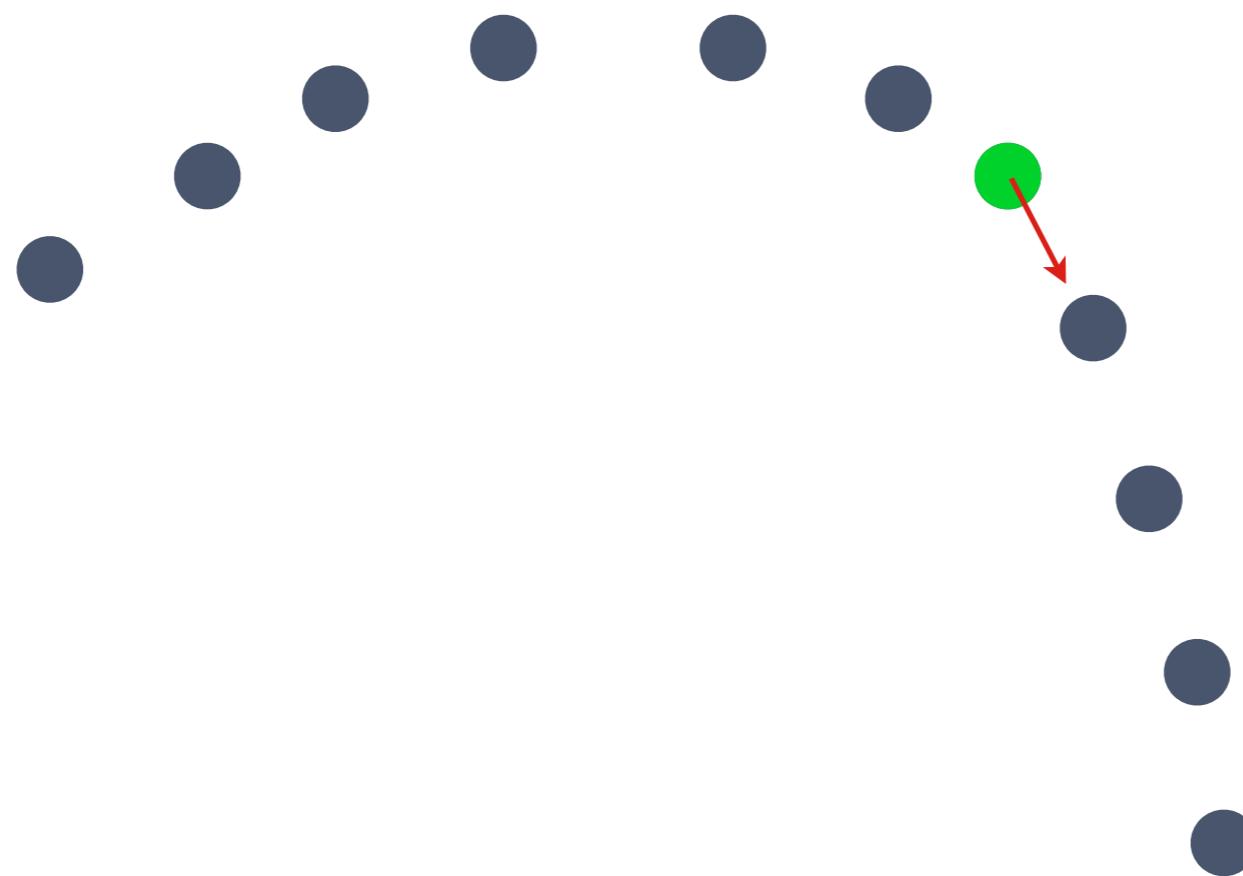
Reconstruction

\Leftarrow tradeoff \Rightarrow

(warning: may require tied weights)

Contraction

pressure to be insensitive
to *all* directions



CAE must capture manifold directions

$$\mathcal{J}_{\text{CAE}} = \sum_{x \in D}^n L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2$$

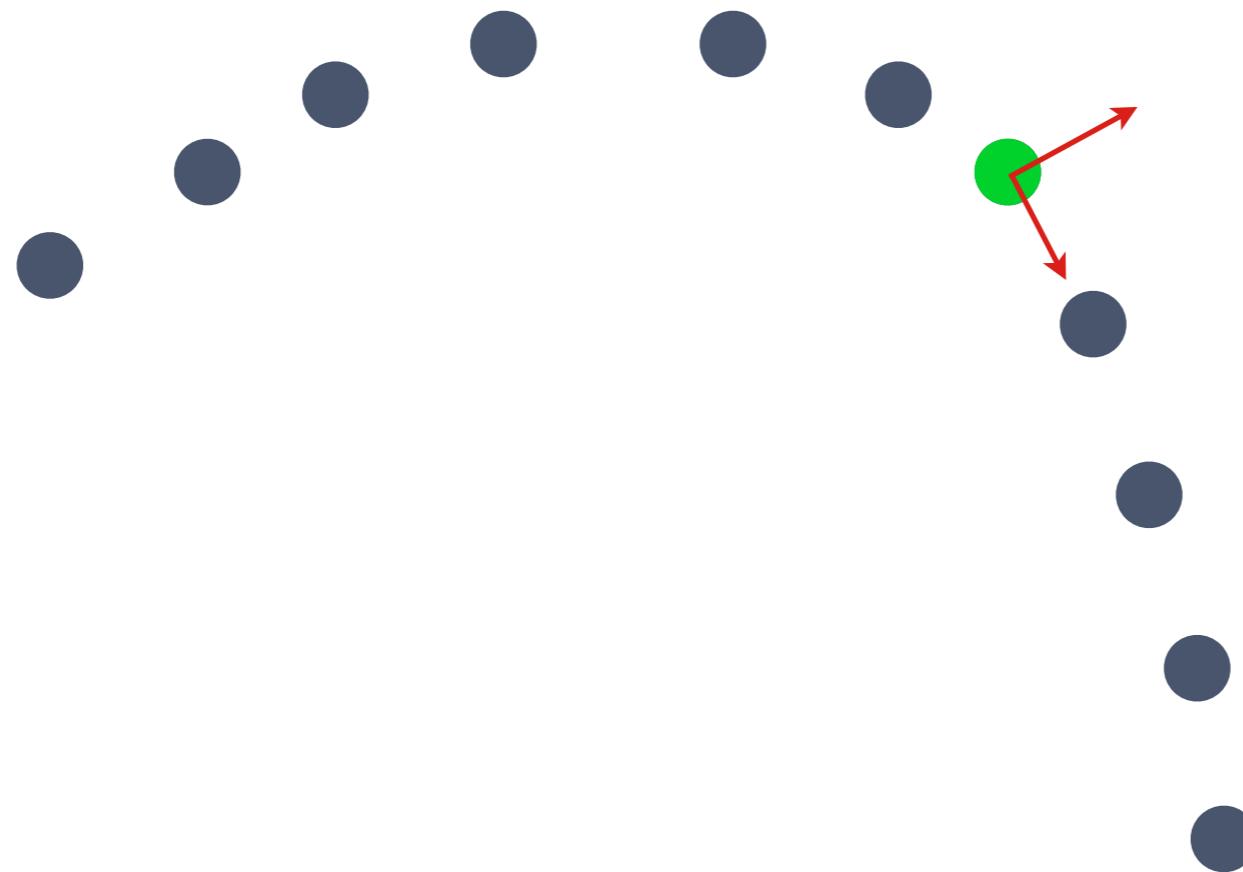
Reconstruction

\Leftarrow tradeoff \Rightarrow

(warning: may require tied weights)

Contraction

pressure to be insensitive
to *all* directions



CAE must capture manifold directions

$$\mathcal{J}_{\text{CAE}} = \sum_{x \in D}^n L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2$$

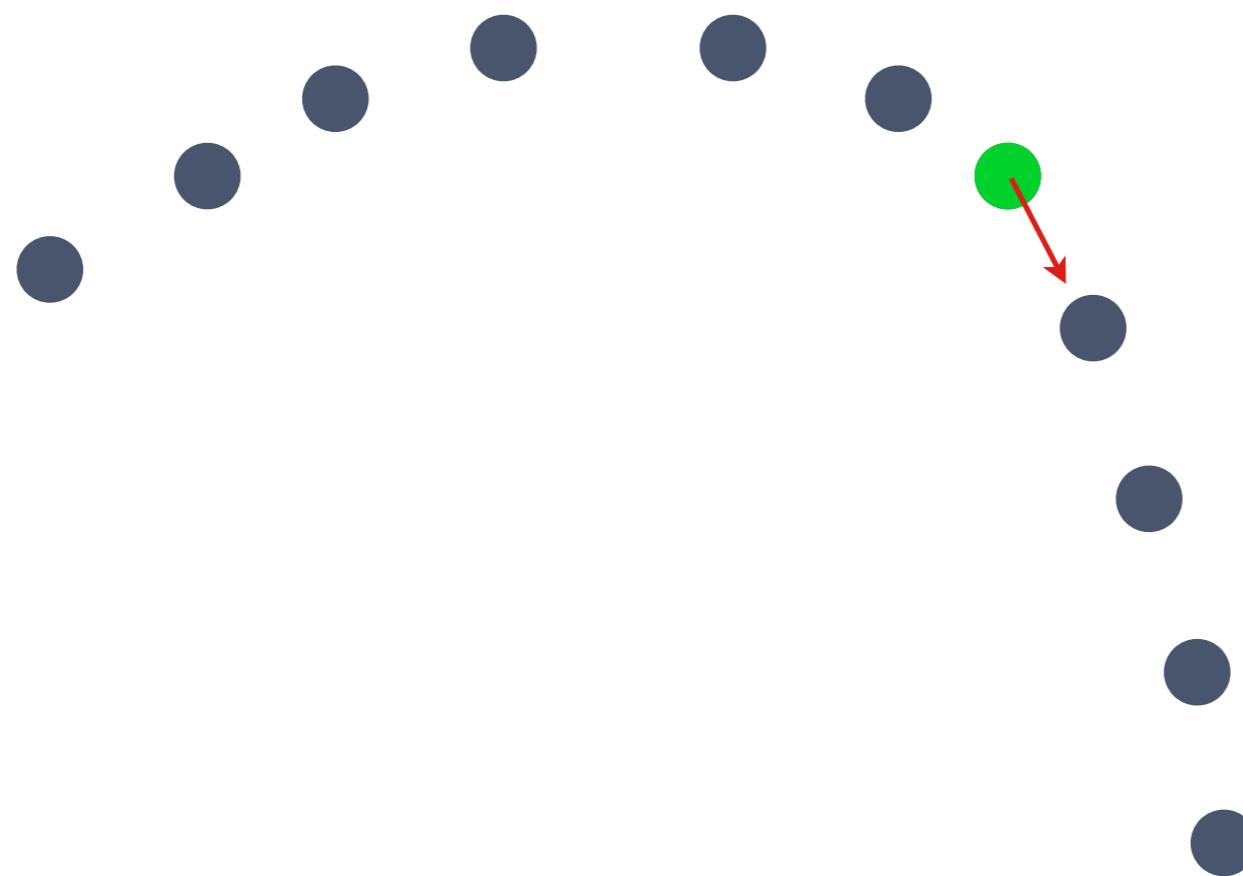
Reconstruction

\Leftarrow tradeoff \Rightarrow

(warning: may require tied weights)

Contraction

pressure to be insensitive
to *all* directions



CAE must capture manifold directions

$$\mathcal{J}_{\text{CAE}} = \sum_{x \in D}^n L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2$$

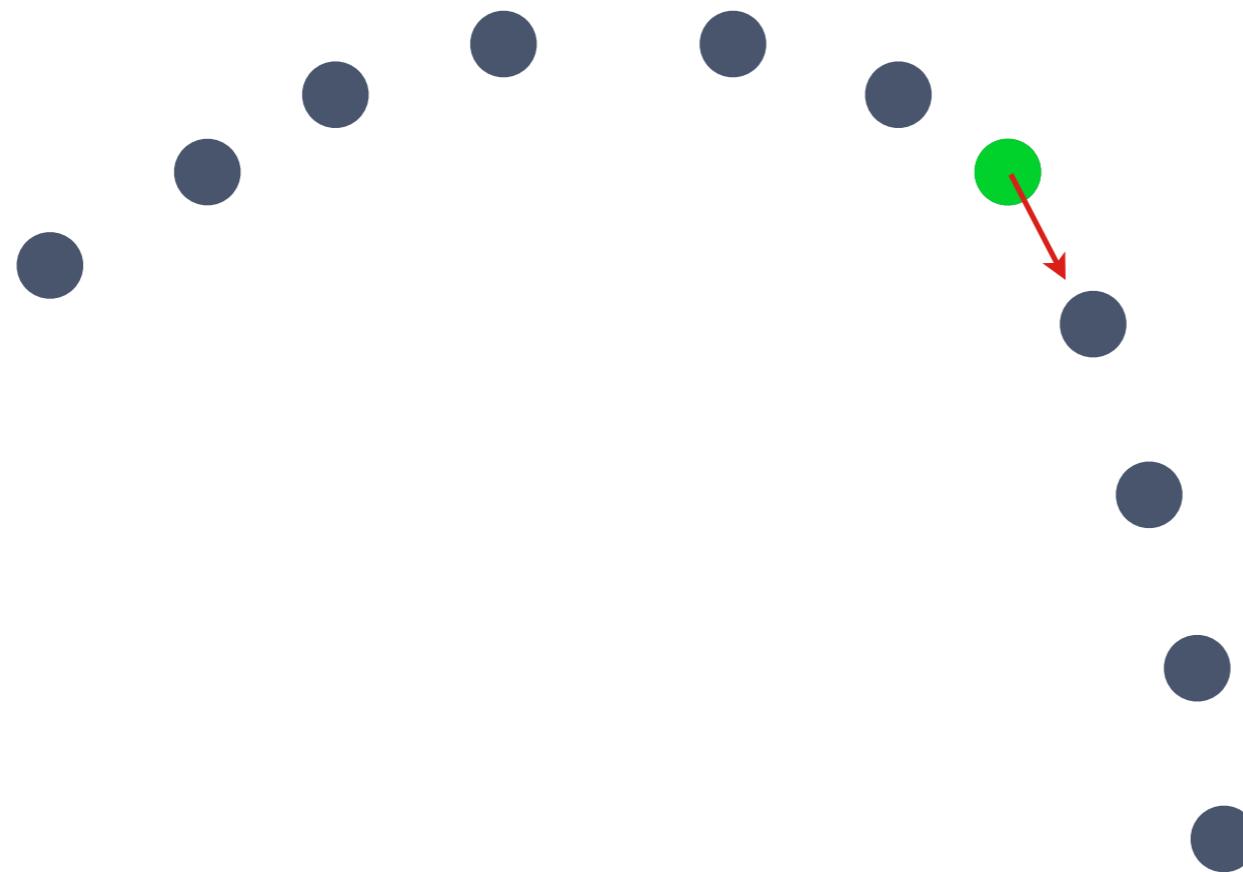
Reconstruction

\Leftarrow tradeoff \Rightarrow

(warning: may require tied weights)

Contraction

pressure to be insensitive
to *all* directions



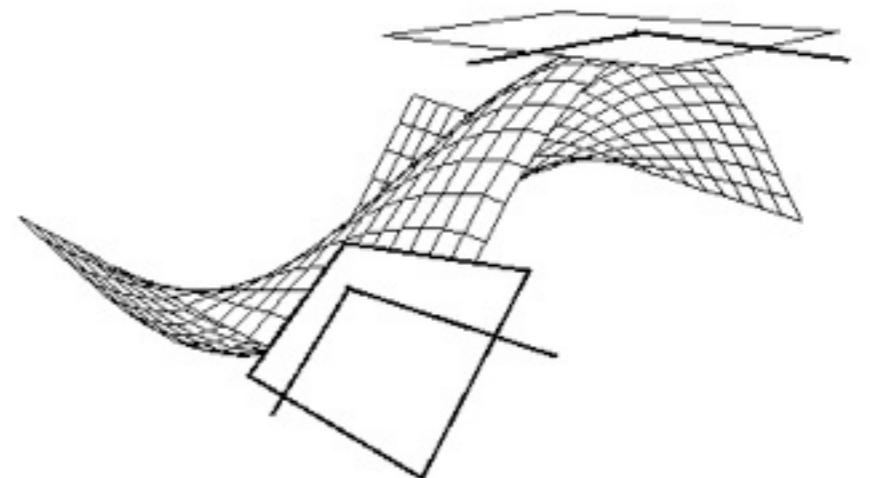
Learned tangent space

-
- Jacobian $J_h(x) = \frac{\partial h}{\partial x}(x)$ measures sensitivity of h locally around x

SVD:

$$\frac{\partial h(\mathbf{x})^T}{\partial \mathbf{x}} = \mathbf{U} \mathbf{S} \mathbf{V}^T$$

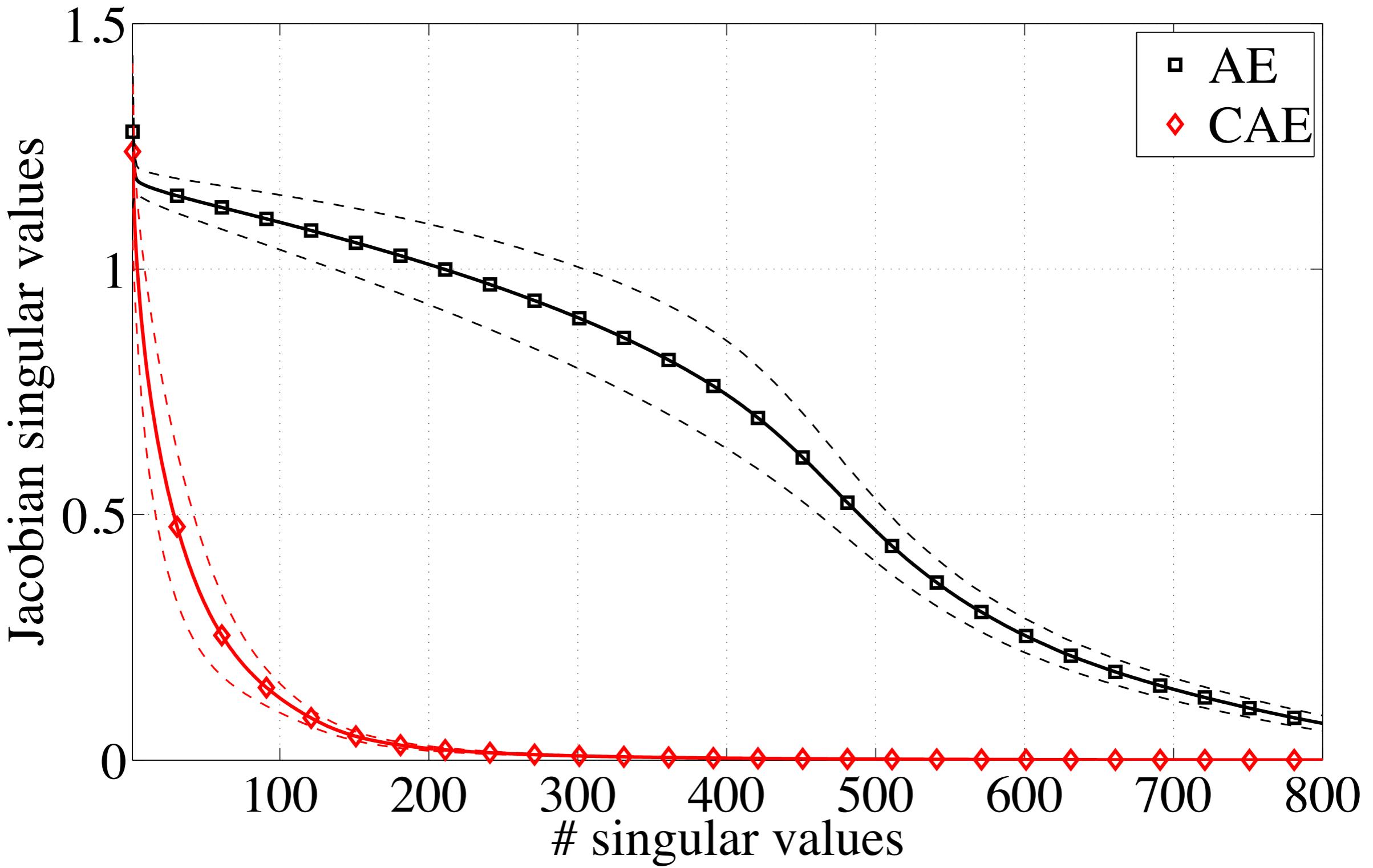
Top singular vectors are tangent directions to which h is most sensitive.



$$\mathbf{T}_x = \{\mathbf{U}_{\cdot k} | \mathbf{S}_{kk} > \epsilon\}$$

SVD of $J_h(x) = \frac{\partial h}{\partial x}(x)$

CIFAR-10

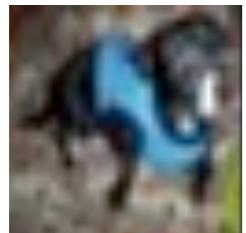


Learned tangents CIFAR-10

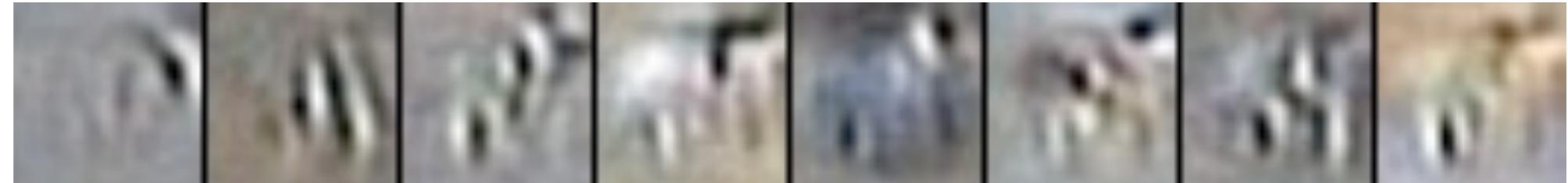
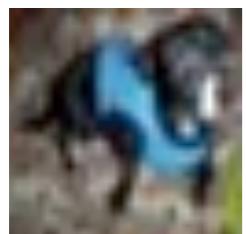
Input Point

Tangents

Local PCA (as e.g. in Manifold Parzen)



Contractive Auto-Encoder (singular vectors of $J_h(x)$)



Not based on explicit neighbors or pairs of points!

Learned tangent spaces

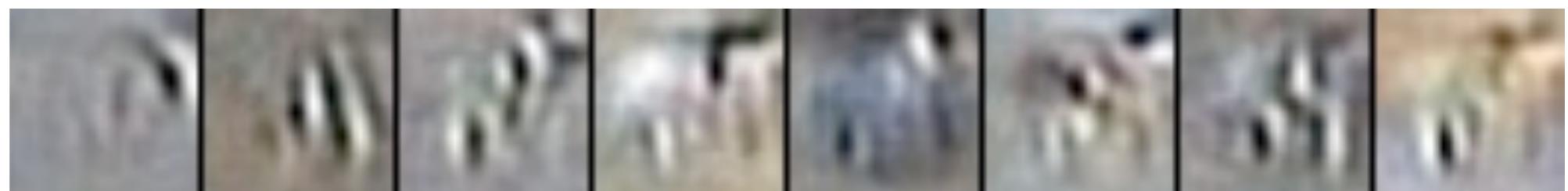
Input Point



Local PCA (as e.g. in



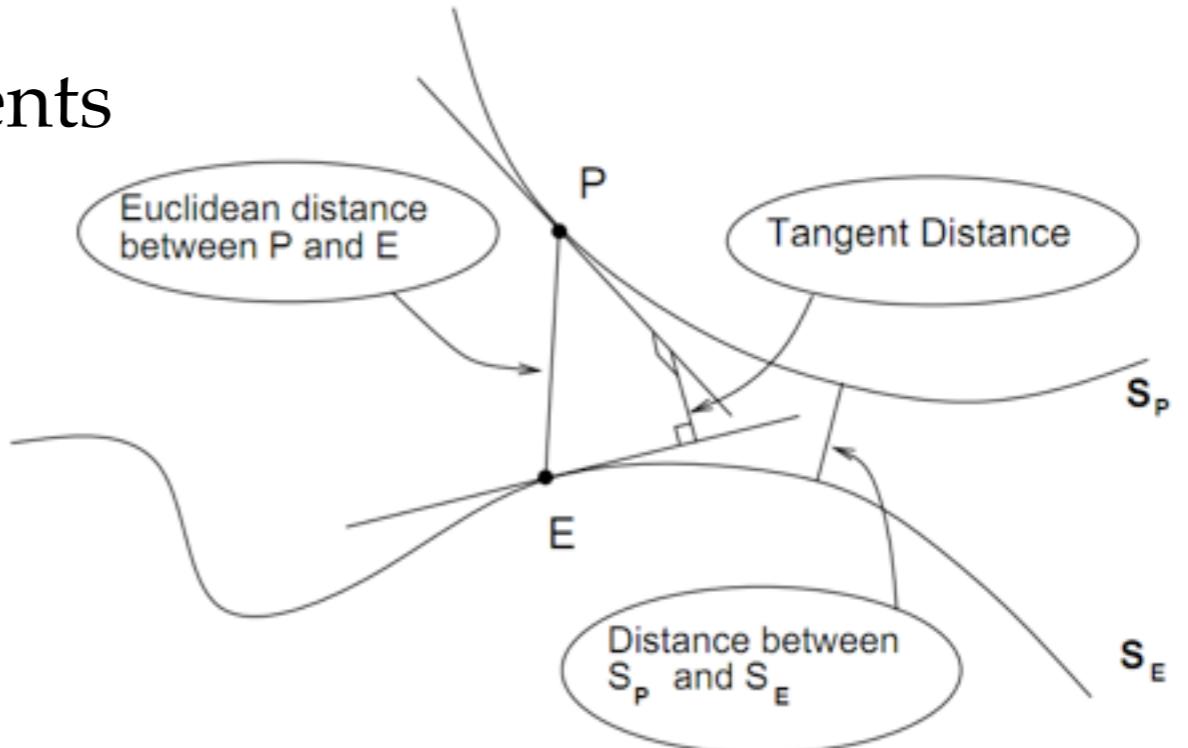
Contractive Auto-Encoder (singular vectors of $J_h(x)$)



Not based on explicit neighbors or pairs of points!

How to leverage the learned tangents

- Simard et al, 1993 exploited tangents derived from prior-knowledge of image deformations **we can use our learned tangents instead.**

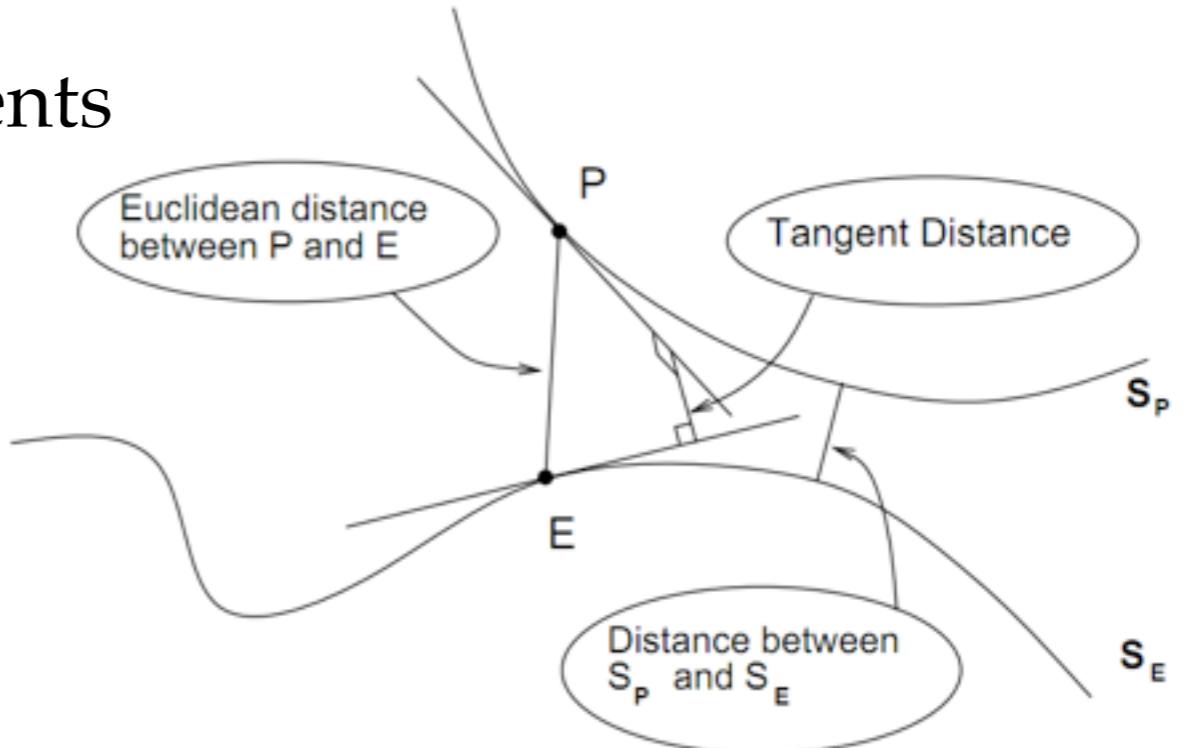


- Use them to define **tangent distance** to use in your favorite distance (k-NN) or kernel-based classifier...
- Use them with **tangent propagation** when fine-tuning a deep-net classifier to make class prediction insensitive to tangent directions. (*Manifold Tangent Classifier*, Rifai et al. NIPS 2011) 0.81% on MNIST
- Moving preferably along tangents allows **efficient quality sampling**



How to leverage the learned tangents

- Simard et al, 1993 exploited tangents derived from prior-knowledge of image deformations **we can use our learned tangents instead.**



- Use them to define **tangent distance** to use in your favorite distance (k-NN) or kernel-based classifier...
- Use them with **tangent propagation** when fine-tuning a deep-net classifier to make class prediction insensitive to tangent directions. (*Manifold Tangent Classifier*, Rifai et al. NIPS 2011) 0.81% on MNIST
- Moving preferably along tangents allows **efficient quality sampling**



Analytic v.s. stochastic ?

a) Analytic approximation of stochastic perturbation

- - Equiv. to tiny perturbations: does not probe far away
- + Potentially more efficient. Ex:
CAE's Jacobian penalty probes sensitivity in all d directions in $O(d_h d)$
With DAE or SCAE it would require encoding d corrupted inputs: $O(d_h d^2)$

b) Stochastic approximation of analytic criterion

- + can render practical otherwise computationally infeasible criteria
Ex: CAE+H
- - less precise, more noisy

CAE+H actually leverages both

Score matching

(Hyvärinen 2005)

We want to **learn** a p.d.f. : $p_\theta(x) = \frac{1}{Z(\theta)} e^{-E_\theta(x)}$
with **intractable** partition function Z

Score matching: alternative **inductive principle** to max. likelihood

Find parameters that minimize objective:

$$J_{SM}(\theta) = \sum_{x \in D} \left(\left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x) \right)$$

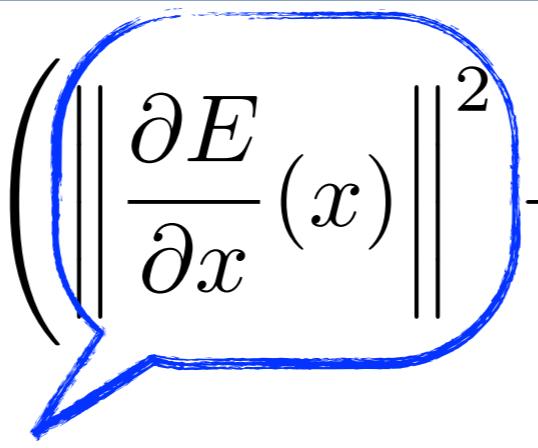
Score matching my geometric interpretation

$$J_{SM}(\theta) = \sum_{x \in D} \left(\left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x) \right)$$

Score matching my geometric interpretation

$$J_{SM}(\theta) = \sum_{x \in D} \left(\left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x) \right)$$

$\|J_E(x)\|^2$



Score matching my geometric interpretation

$$J_{SM}(\theta) = \sum_{x \in D} \left(\left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x) \right)$$

$\|J_E(x)\|^2$

First derivative encouraged to be small: ensures training points stay close to local minima of E

Score matching my geometric interpretation

$$J_{SM}(\theta) = \sum_{x \in D} \left(\left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x) \right)$$

$\|J_E(x)\|^2$ Laplacian $\text{Tr}(H_E(x))$

First derivative encouraged to be small: ensures training points stay close to local minima of E

Score matching my geometric interpretation

$$J_{SM}(\theta) = \sum_{x \in D} \left(\left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x) \right)$$

$\|J_E(x)\|^2$ Laplacian $\text{Tr}(H_E(x))$

First derivative encouraged to be small: ensures training points stay close to local minima of E

Encourage large positive curvature in all directions

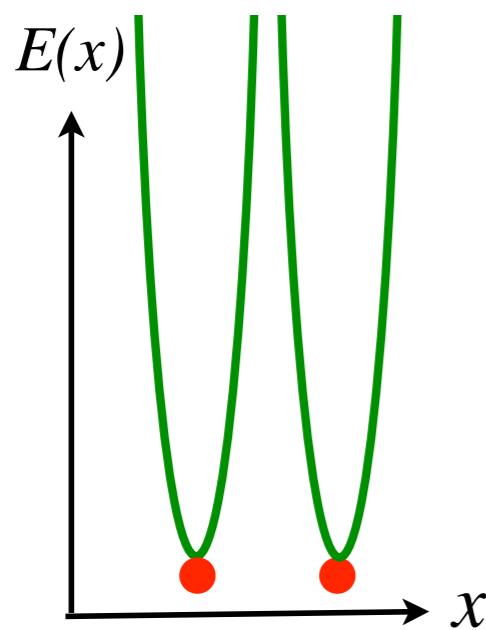
Score matching my geometric interpretation

$$J_{SM}(\theta) = \sum_{x \in D} \left(\left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x) \right)$$

$\|J_E(x)\|^2$ Laplacian $\text{Tr}(H_E(x))$

First derivative encouraged to be small: ensures training points stay close to local minima of E

Encourage large positive curvature in all directions



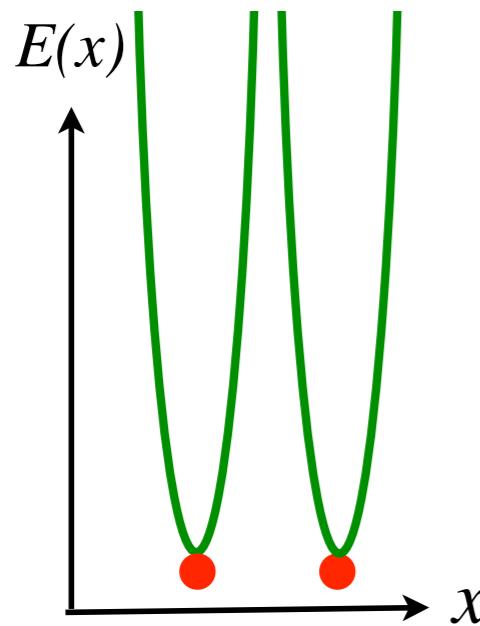
Score matching my geometric interpretation

$$J_{SM}(\theta) = \sum_{x \in D} \left(\left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x) \right)$$

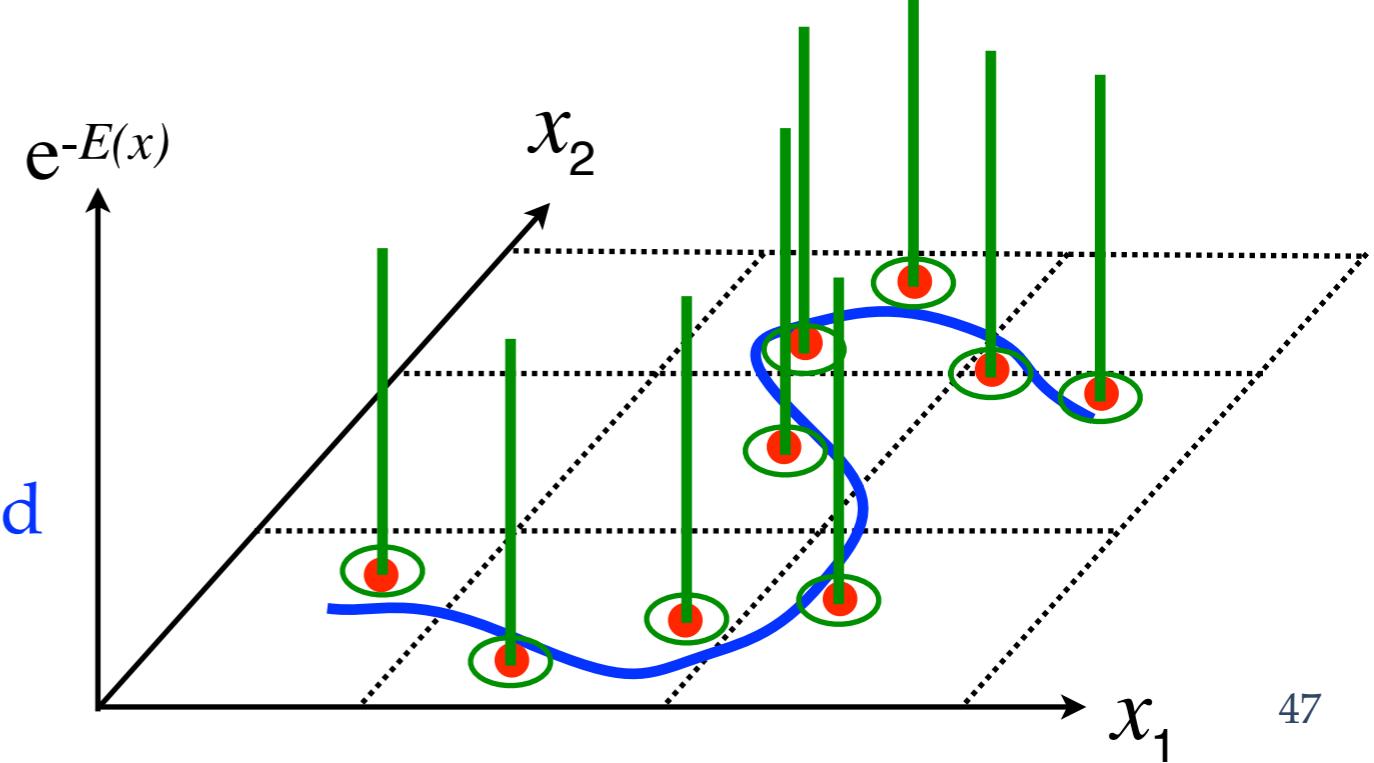
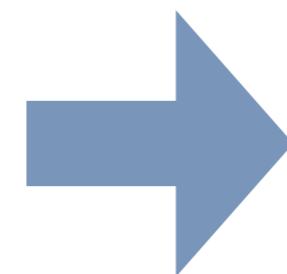
|| $J_E(x)$ ||²
Laplacian
Tr($H_E(x)$)

First derivative encouraged to be small: ensures training points stay close to local minima of E

Encourage large positive curvature in all directions



sharply peaked density



Score matching variants

Original score matching (Hyvärinen 2005):

$$J_{SM}(\theta) = \sum_{x \in D} \left(\frac{1}{2} \left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x) \right)$$

Analytic

Regularized score matching (Kingma & LeCun 2010):

$$J_{SMreg,\lambda}(\theta) = J_{SM} + \sum_{x \in D} \lambda \sum_{i=1}^d \frac{\partial^2 E}{\partial x_i^2}(x)$$

Analytic

Denoising score matching (Vincent 2011)

$$J_{DSM,\sigma} = \sum_{x \in D} \left(\mathbf{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 I)} \left[\frac{1}{2} \left\| \frac{\partial E}{\partial x}(x + \epsilon) - \frac{1}{\sigma^2} \epsilon \right\|^2 \right] \right)$$

Stochastic

DAE training has a deeper relationship to RBMs

- Same functional form as RBM:
 $h(x)$ is expected hidden given visible
 $g(\mathbf{h})$ is expected visible given hidden
- With linear reconstruction and squared error,
DAE amounts to learning the following energy

$$E(\mathbf{x}; \underbrace{\mathbf{W}, \mathbf{b}, \mathbf{c}}_{\theta}) = -\frac{\langle \mathbf{c}, \mathbf{x} \rangle - \frac{1}{2} \|\mathbf{x}\|^2 + \sum_{j=1}^{d_h} \text{softplus}(\langle \mathbf{W}_j, \mathbf{x} \rangle + \mathbf{b}_j)}{\sigma^2}$$

using the denoising score matching inductive principle.

- Above energy closely related to free energy of Gaussian-binary RBM (identical for $\sigma=1$)



Questions ?