Representation of High Dimensional Data

Ross Goroshin

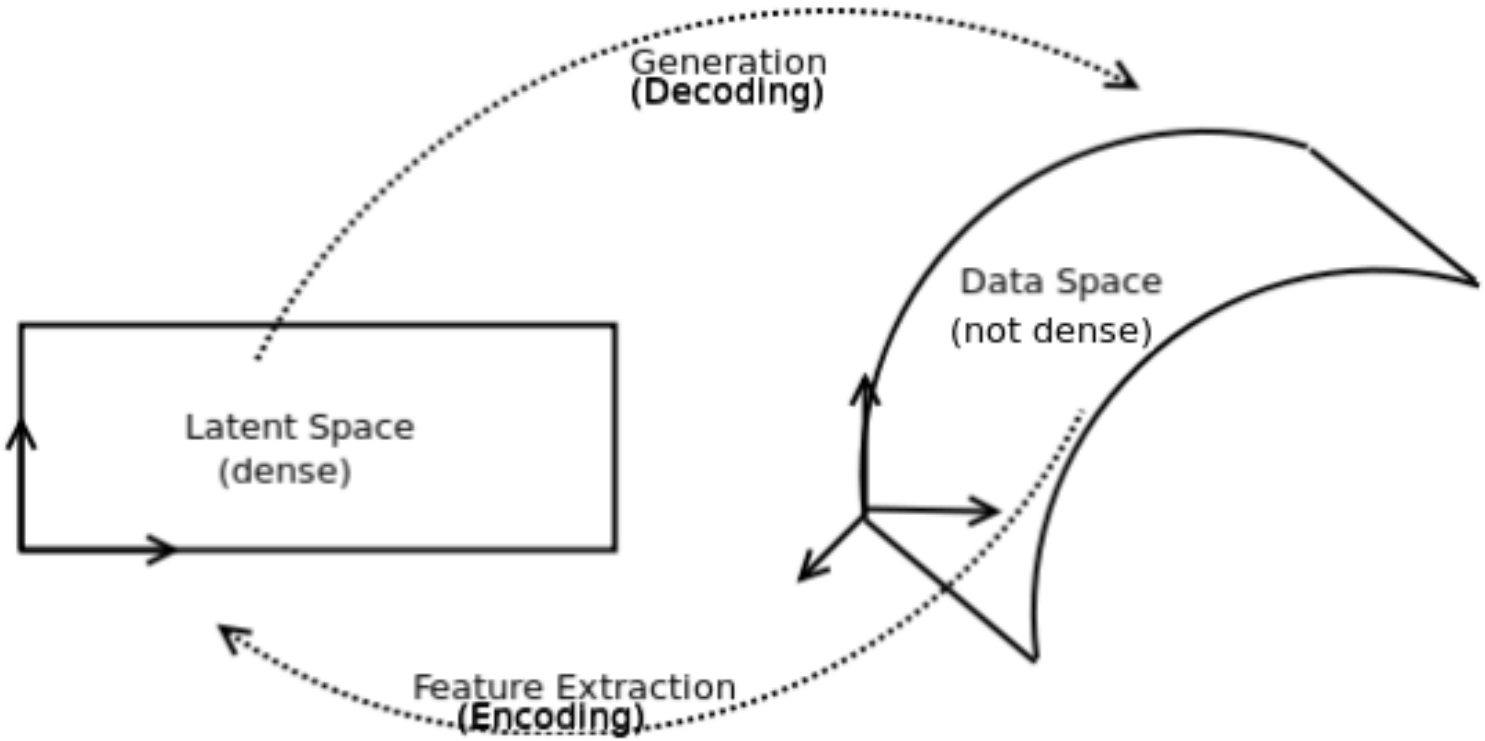
October 30, 2012

NEW YORK UNIVERSITY

Dimensionality of Data & Statistical Dependence

- Suppose we have a 42 second video played at 24 frames/second, with a resolution of 1000 by 1000 pixels
- In theory each pixel can vary independently from frame to frame, which implies that there are $\approx 10^9$ degrees of freedom
- If all of these pixels were to vary independently of one another, the picture would not be very interesting
- Moreover if you were to describe the content of the video to a friend over the telephone, it is doubtful that the term 'pixel' would ever be mentioned in the conversation

Dimensionality of Data & Statistical Dependence



- This illustration is representative of many processes
- However, dependence can be introduced without increasing the dimensionality
- Latent representation is NOT unique for generative processes of interest

Relationship Between Approaches

Algorithm	Model	Encode	Decode	Relate Enc. & Dec.
PCA	Global Linear	✓	✓	$W_D = W_E^T$
ICA	Global Linear	✓	✓	$W_D = W_E^T$
Sparse Coding	Local Linear	✓($)	✓	$W_D = W_E^T$
PSD & LISTA	Local Linear	✓	✓	Learned W_E
DrLIM	Nonlinear	✓	X	Enc. Only
Auto-Encoders	Nonlinear	✓	✓	Learned W_E & W_D

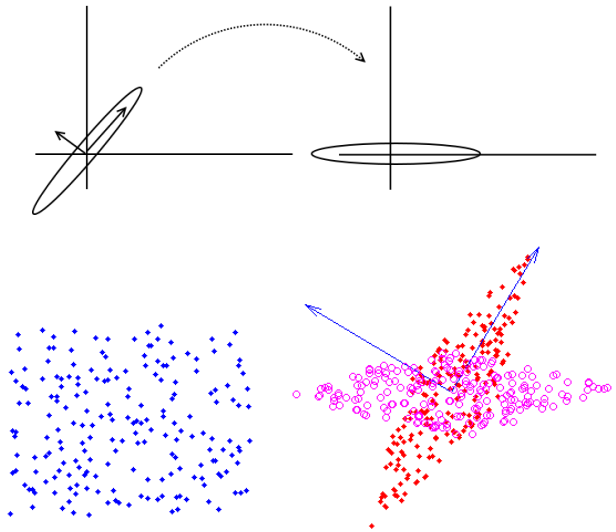
	De-correlation/Independence
	Sparsity
	Metric Learning/Restricted Metric Learning
	All of the Above

First Attempt at Independence: PCA

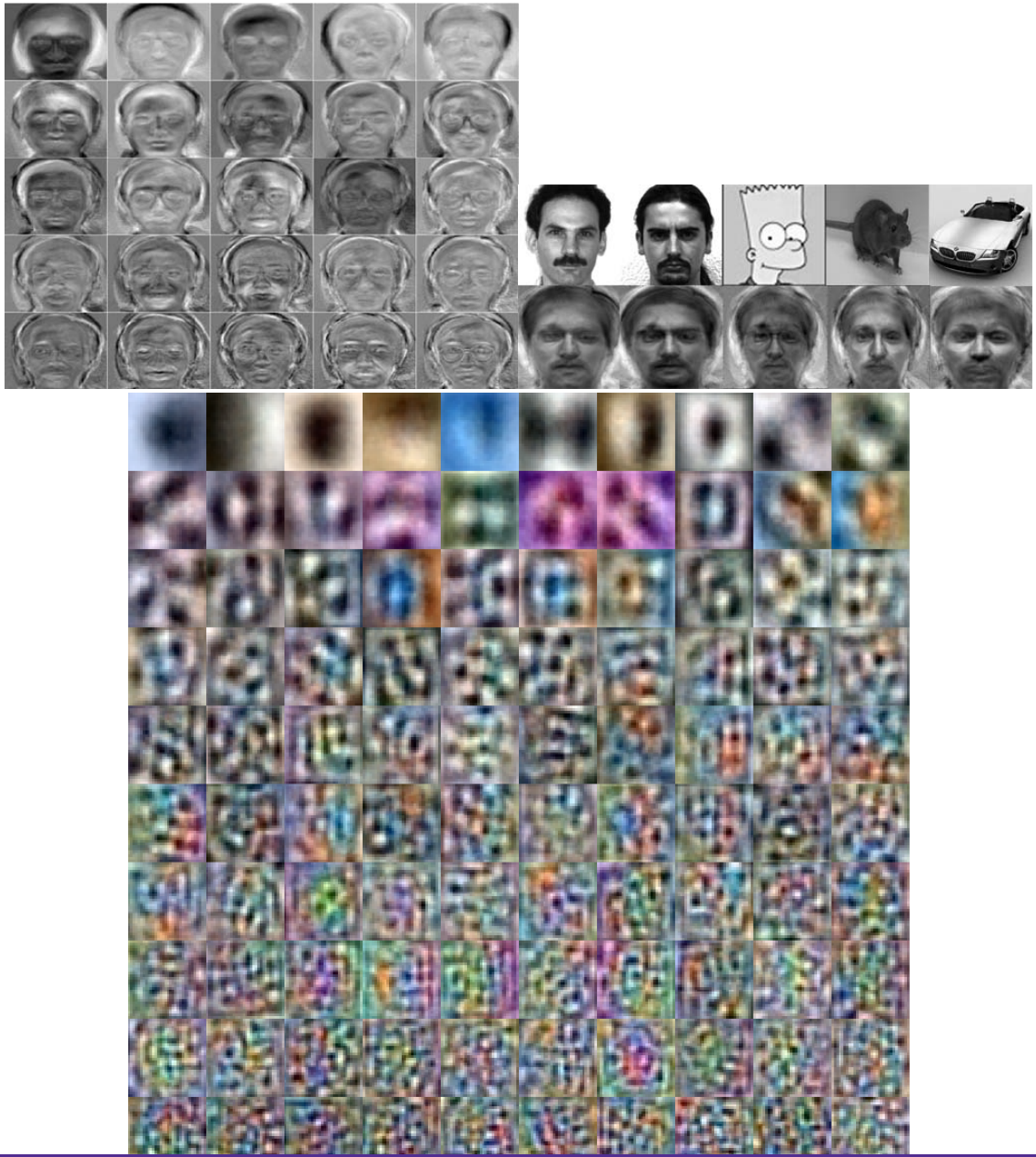
- Assume you have $x = As$ where each $x_i \in \mathbb{R}^D$ (s blue, x red)
- PCA assumes that there are $M \leq D$ linearly interdependent combinations of the input space variables which are responsible for most of the variance of the data

$$\frac{1}{N}\sum_{n=1}^{N}(e_1^T x_n - e_1^T \bar{x})^2 = e_1^T \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T e_1$$

Leading to the problem: maximize $e_1^T \Sigma e_1$ s.t $e_1^T e_1 = 1$ where Σ is the covariance matrix.



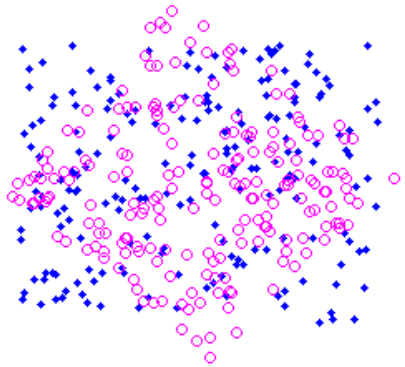
Global Variations are Not Everything



A Little Closer to Independence: Whitening

- Note that the covariance matrix of the data in PCA space is diagonal, i.e. the data is completely uncorrelated
- Whitening the data is equalizing the variance of the uncorrelated data
- The whitening transform is given by $V = WD^{-1/2}W^T$
- The whiteness property of the data is invariant to orthogonal transforms. Let $E[zz^T] = \mathbf{I}$, and let $y = Uz$ where U is an orthogonal transform.
- Then $E[yy^T] = E[Uzz^T U^T] = UE[zz^T]U^T = \mathbf{I}$. Whitening produces the ICs, modulo an orthogonal transform

For radially symmetric distributions (e.g. Gaussian) we are done!

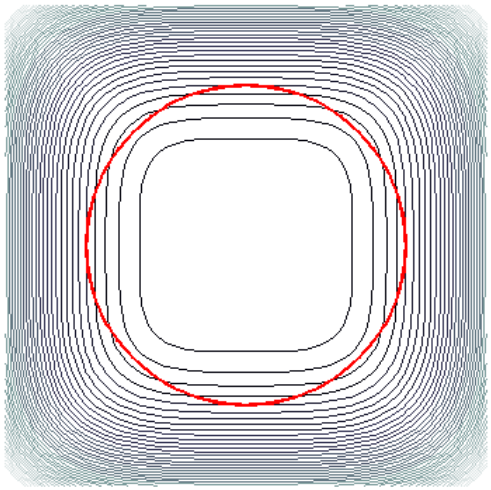


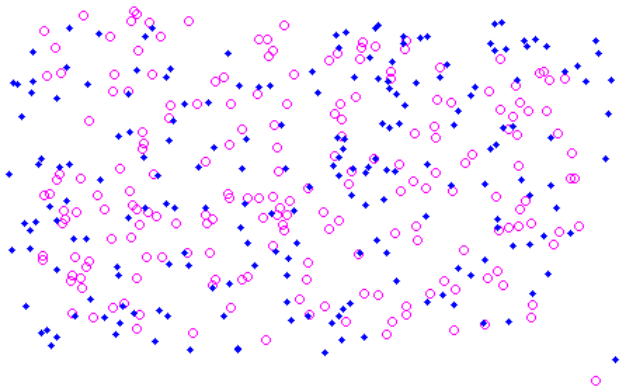
Independent Component Analysis

- Generative model $x = As$, where neither the A nor the s are known. Focus on picking out one of the s components at a time, i.e. $y = b^T As (= q_1 s_1 + q_2 s_2$ for 2D)
- Assume that the s mixture components are i.i.d and non-Gaussian, by the central limit theorem any mixture of the variables is more "Gaussian" than the individual distributions
- A good measure of non-Gaussianity is $k(y) = E[y^4] - 3(E[y^2])^2$. For whitened data, $k(y) = E[y^4] - 3$
- Since the data has been whitened, we constrain $E[y^2] = q_1^2 var(s_1) + q_2^2 var(s_2) + cov(s_1, s_2) = q_1^2 + q_2^2 = 1$

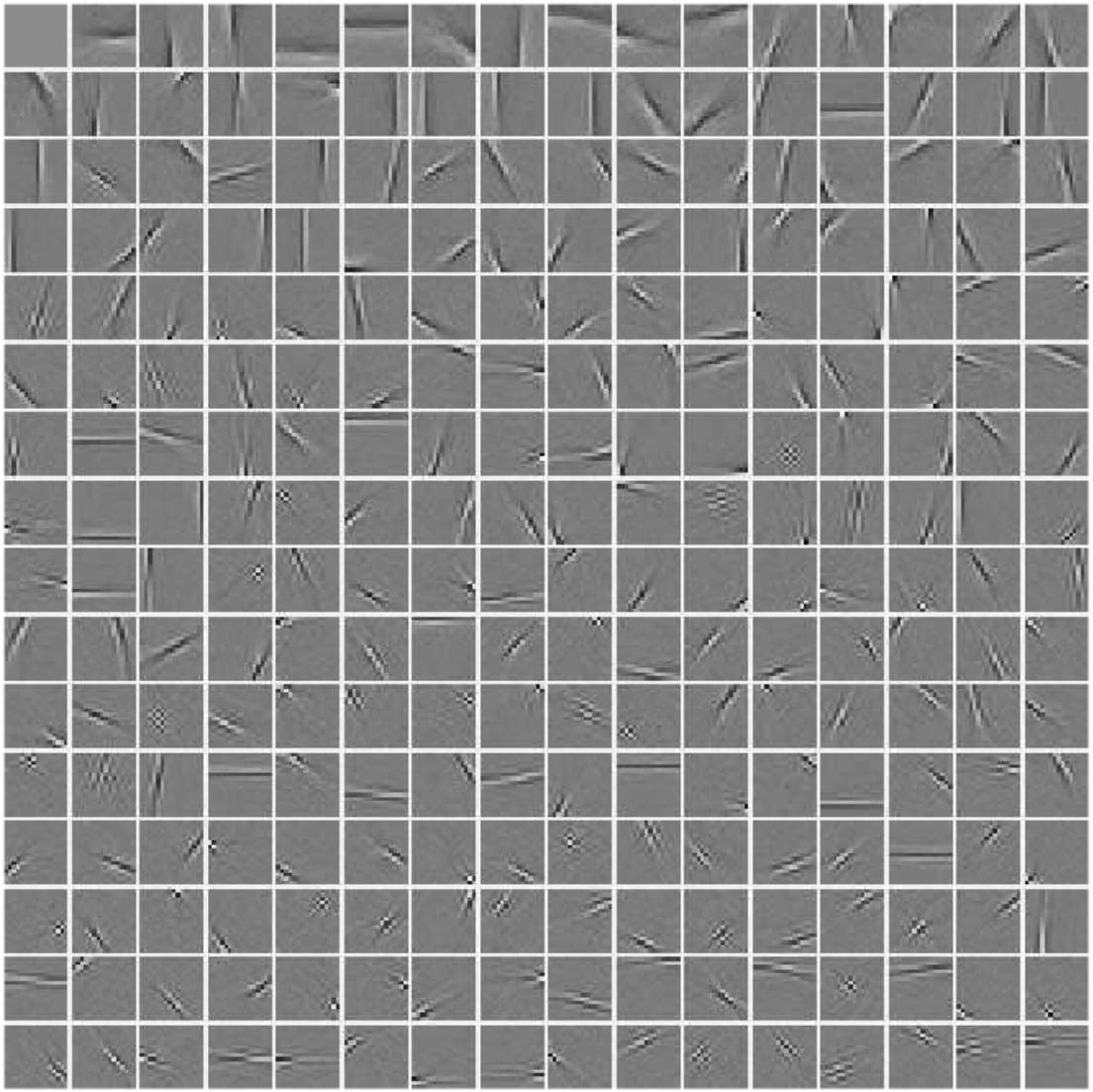
Independent Component Analysis

- $\max |kurt(y)| = |q_1^4 kurt(s_1) + q_2^4 kurt(s_2)|$ s.t. $q_1^2 + q_2^2 = 1$
- Assuming that s_1 and s_2 are i.i.d then $kurt(s_1) = kurt(s_2)$.
- However we don't directly have access to the q variables (they are mixed by A) making it expensive to enforce the constraint $\|q\|^2 = 1$. If the data is whitened however, and we seek the linear combination $w^T z$ that maximizes non-Gaussianity then it can be show that $\|q\| = \|w\|$

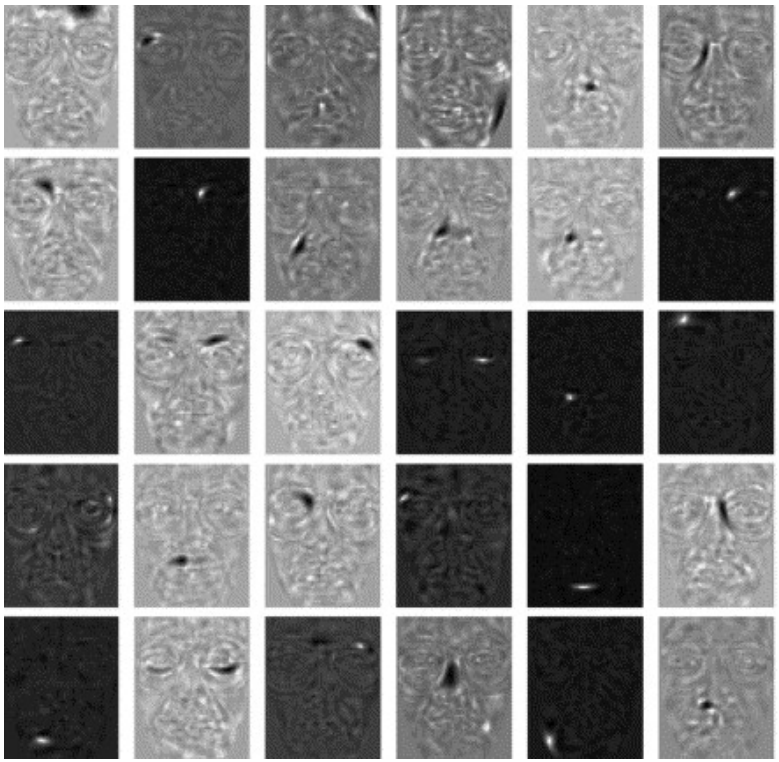




ICA Results



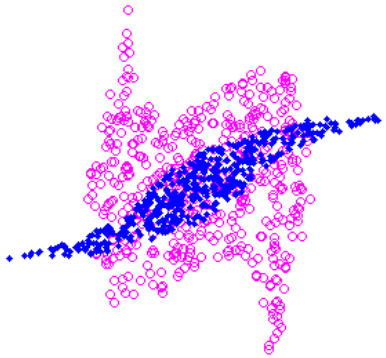
ICA Results



- Note that vectors found by ICA are more localized, i.e. individual activations are sparsely distributed (super-Gaussian) over the data

Limit of Classical ICA

The $x = As$ model is invalid for nonlinear data manifolds



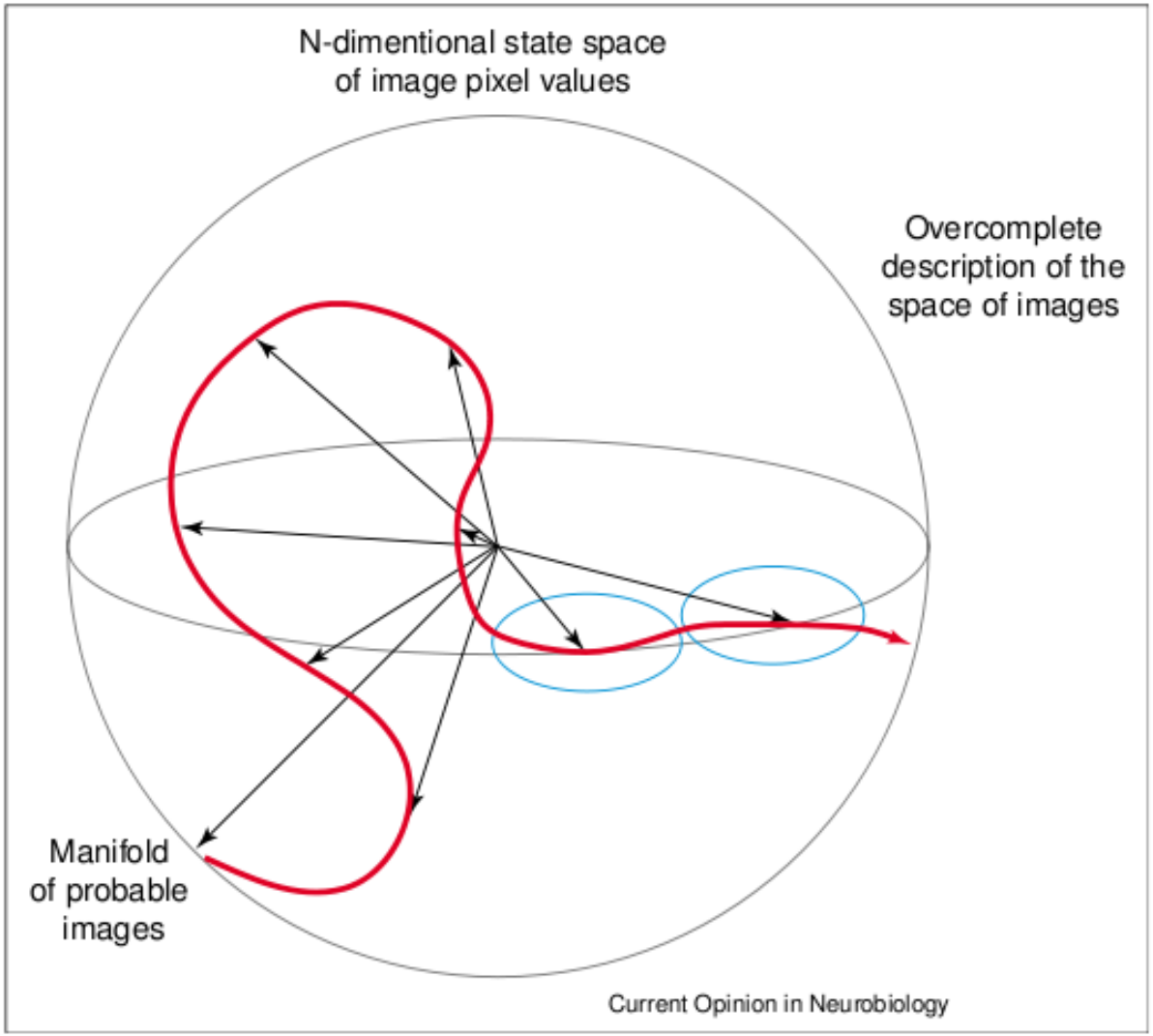
- One way to approximate a nonlinear independent direction is to use a local linear approximation, which requires an over-complete basis
- Heuristic requirements for independence: Basis vectors should be sparsely activated

Relationship Between Approaches

Algorithm	Model	Encode	Decode	Relate Enc. & Dec.
PCA	Global Linear	✓	✓	$W_D = W_E^T$
ICA	Global Linear	✓	✓	$W_D = W_E^T$
Sparse Coding	Local Linear	✓($)	✓	$W_D = W_E^T$
PSD & LISTA	Local Linear	✓	✓	Learned W_E
DrLIM	Nonlinear	✓	X	Enc. Only
Auto-Encoders	Nonlinear	✓	✓	Learned W_E & W_D

	De-correlation/Independence
	Sparsity
	Metric Learning/Restricted Metric Learning
	All of the Above

Sparse Coding



Unlike local PCA, basis elements can be recycled

Sparse Coding

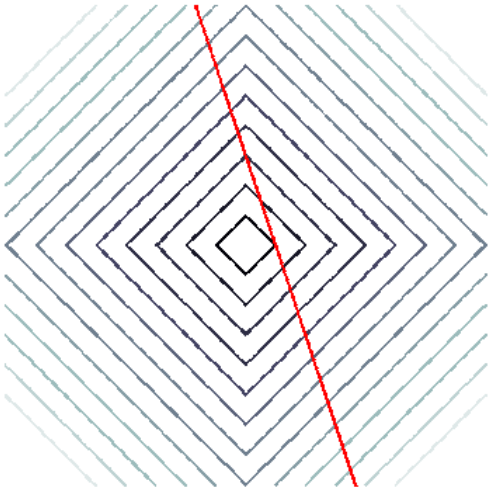
- Find a sparsely activated over-complete basis which describes the data
- Direct measure of sparsity, the L_0 norm, results in a combinatorial optimization problem
- Note that another popular measure of sparsity is kurtosis, which links directly back to ICA
- Modern formulations of sparse coding use the L_1 norm as a sparsity measure

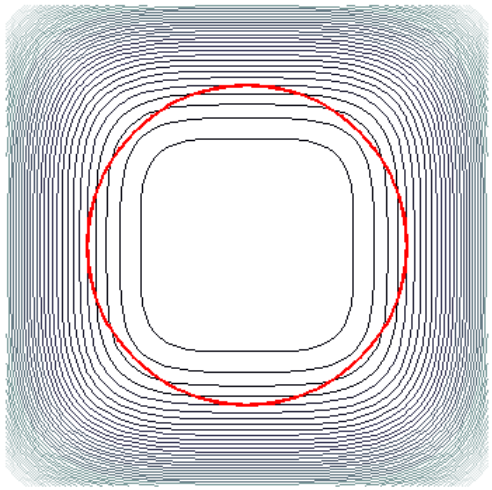
$$min\|z\|_1 \text{ s.t } Wz = s \text{ (BP)}$$

$$\min_z \frac{1}{2}\|s - Wz\|_2^2 + \lambda\|z\|_1 \text{ (BPDN)}$$

$$\min_{z,W} \frac{1}{2}\|s - Wz\|_2^2 + \lambda\|z\|_1 \text{ (SC)}$$

Why L_1?





Left: Minimize L_1, goal: sparsity
Right: Maximize kurotsis, goal: independence

Implementation

$$\min_{z,W} \frac{1}{2}\|s - Wz\|_2^2 + \lambda\|z\|_1$$

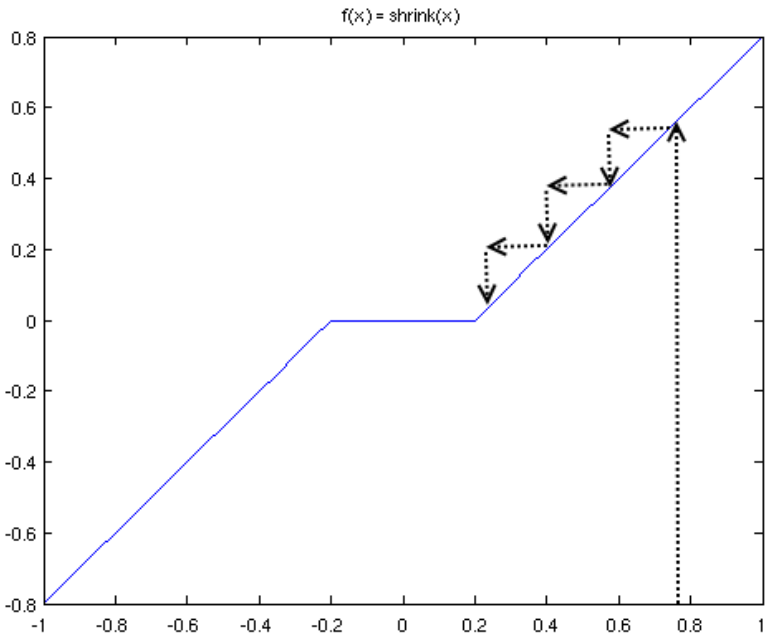
- Alternately optimize z (inference) and W (basis update)
- It is easy to reduce $\|z\|_1$ and increase the norm of the columns of W, thus the columns of W must be normalized to unity

ISTA Inference

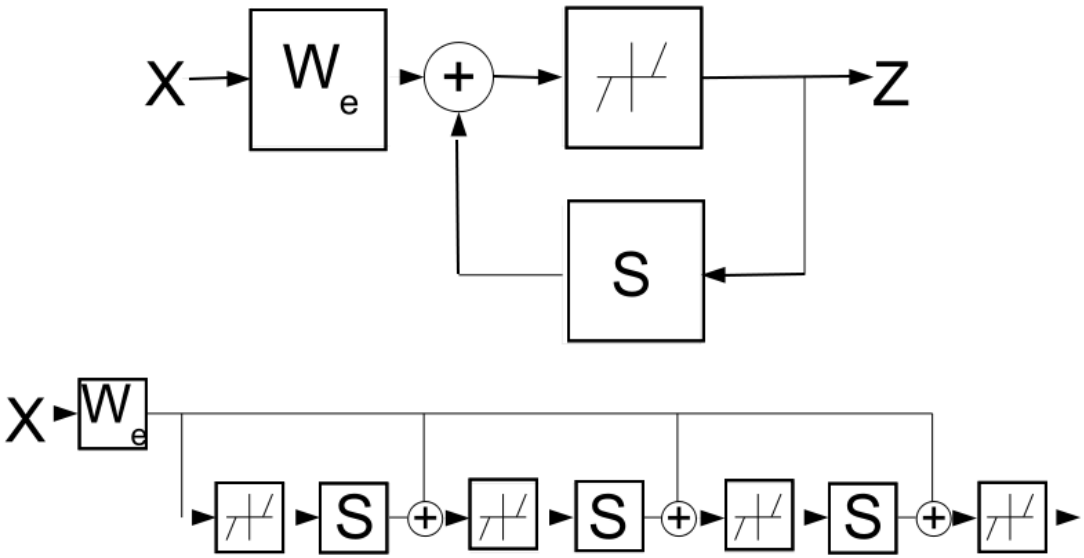
- Given a fixed basis W there exists a fixed point algorithm for finding the optimal coefficients z^* (i.e. inference)

$$z_{k+1} = shrink(z_k - \eta_1 \nabla_{z_k} \frac{1}{2} \|s - Wz_k\|_2^2)$$

- Application of the $shrink()$ function corresponds to a gradient step in L_1



LISTA Inference



$$S = I - \frac{1}{L} W_d^T W_d \text{ and } W_e = \frac{1}{L} W_d^T$$

- Learned ISTA is a method for approximating the inference step by a feed forward network
- Since only a finite number of ISTA iterations are performed, W_e and S are trainable
- Allows the possibility of 'explaining away', unlike PSD

Invariance and IPSD

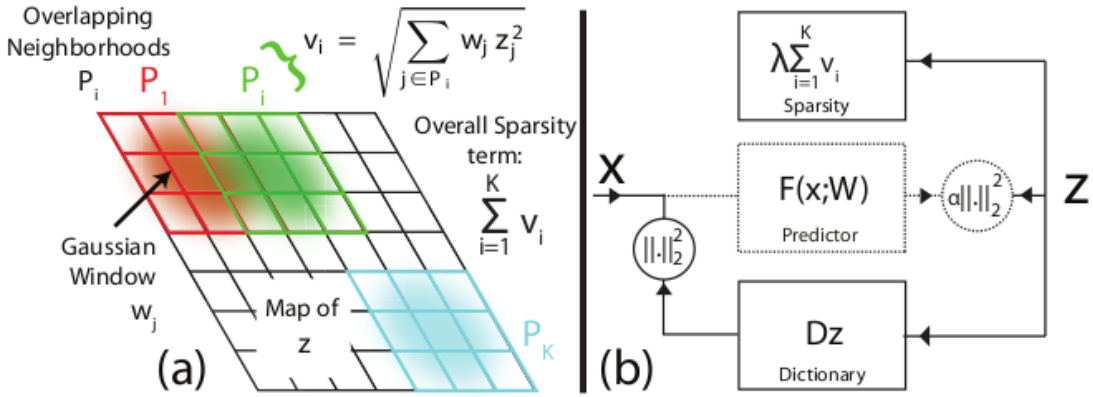
- For a sparse code $\|\frac{\partial z}{\partial x}\|$ can be huge (discontinuous)
- Invariance is a desired quality, ex: SIFT, HoG
- Approach: Imposed a topological structure on z by pooling over overlapping neighborhoods, and measure sparsity of the neighborhoods

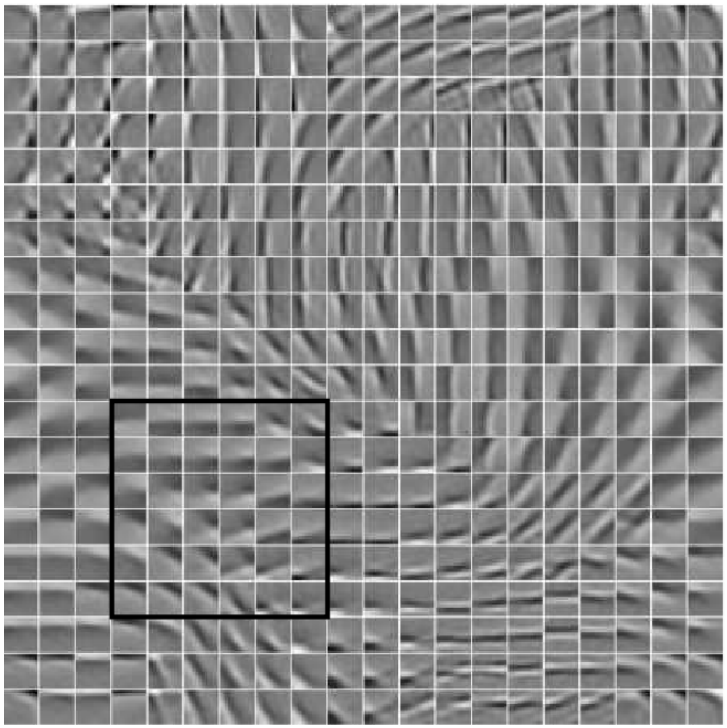
$$\frac{1}{2}\|x - Wz\|_2^2 + \lambda \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i} w_j z_j^2}$$

- $\sqrt{n} < n$ for $n > 1$, encourages sparse activations across neighborhoods
- Within neighborhoods z_j are encouraged to co-activate

$$\frac{1}{2}\|x - Wz\|_2^2 + \lambda \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i} w_j z_j^2} + \alpha\|z - F(x; W)\|_2^2$$

Invariance and IPSD





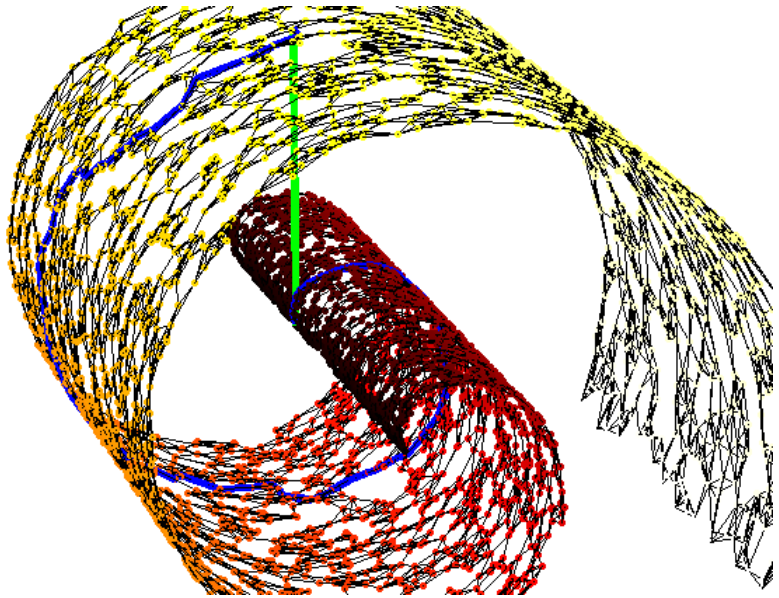
Relationship Between Approaches

Algorithm	Model	Encode	Decode	Relate Enc. & Dec.
PCA	Global Linear	✓	✓	$W_D = W_E^T$
ICA	Global Linear	✓	✓	$W_D = W_E^T$
Sparse Coding	Local Linear	✓($)	✓	$W_D = W_E^T$
PSD & LISTA	Local Linear	✓	✓	Learned W_E
DrLIM	Nonlinear	✓	X	Enc. Only
Auto-Encoders	Nonlinear	✓	✓	Learned W_E & W_D

	De-correlation/Independence
	Sparsity
	Metric Learning/Restricted Metric Learning
	All of the Above

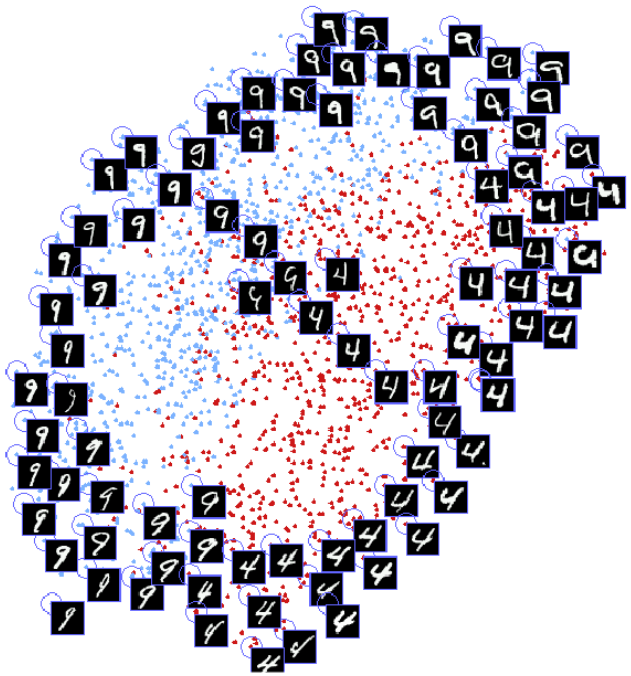
Metric Learning

- Geodesic v.s. Euclidean distance
- Another way to pose the problem is to find a distance preserving mapping to lower dimensional space
- For densely sampled manifolds this makes sense, but for realistic data we are satisfied with preserving some distance metric of interest, possibly mangling others



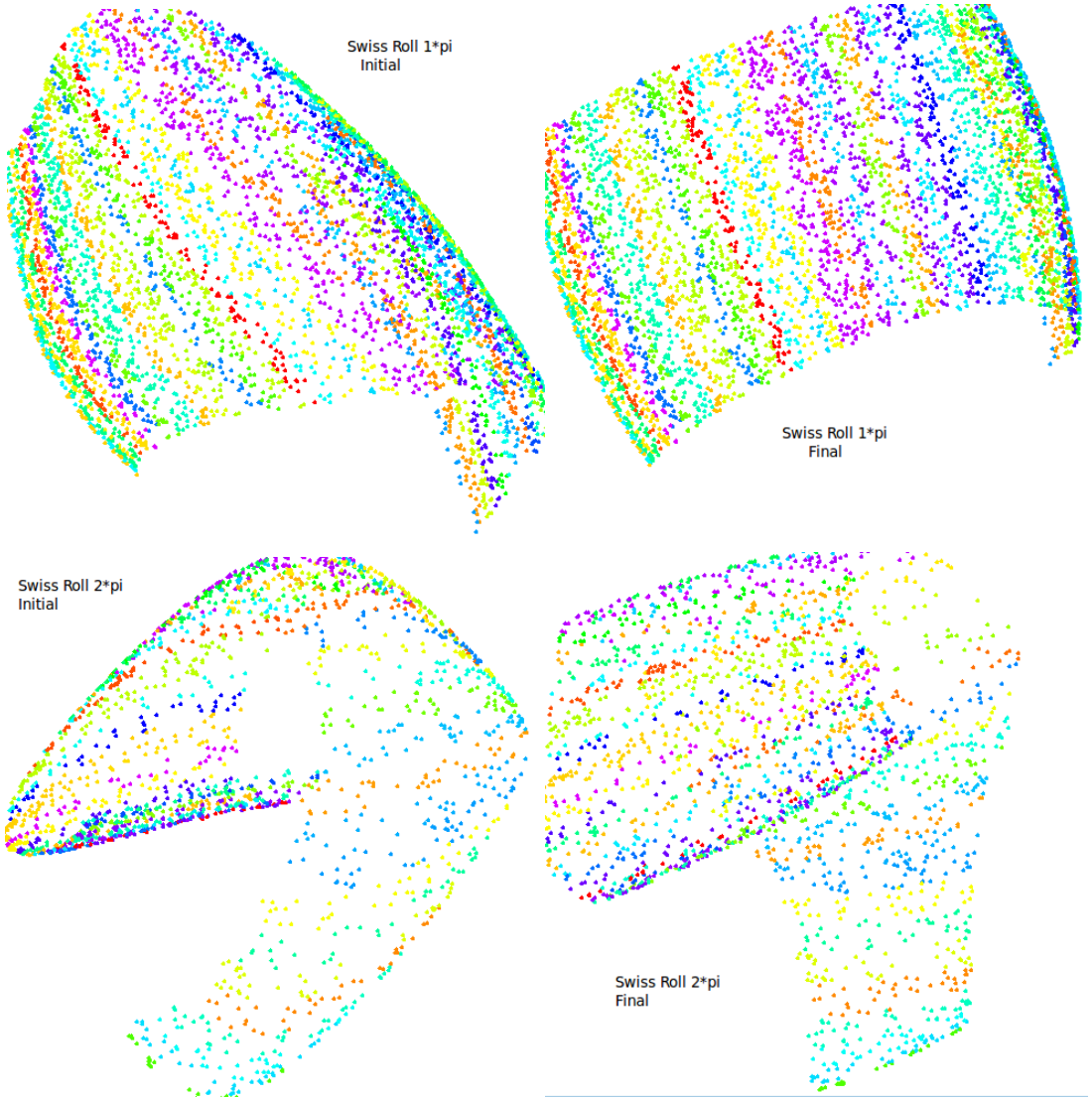
DrLIM

- We wish to find a mapping $G_W(X_i) : \mathbb{R}^D \rightarrow \mathbb{R}^d$, where $D > d$ which translates labeled similarity relationships in the input space to Euclidean distances in the output space
- If (X_1, X_2) are similar then $Y = 0$, otherwise $Y = 1$
- Let $D_W(X_1, X_2) = \|G_W(X_1), G_W(X_2)\|_2$
- $L(W, Y, X_1, X_2) = (1 - Y)\frac{1}{2}D_W^2 + Y\frac{1}{2}\{max(0, m - D_W)\}^2$



DrLIM

We can do classical metric learning using DrLIM by using L_2 as a measure of similarity in the ambient space (n-nearest neighbors)



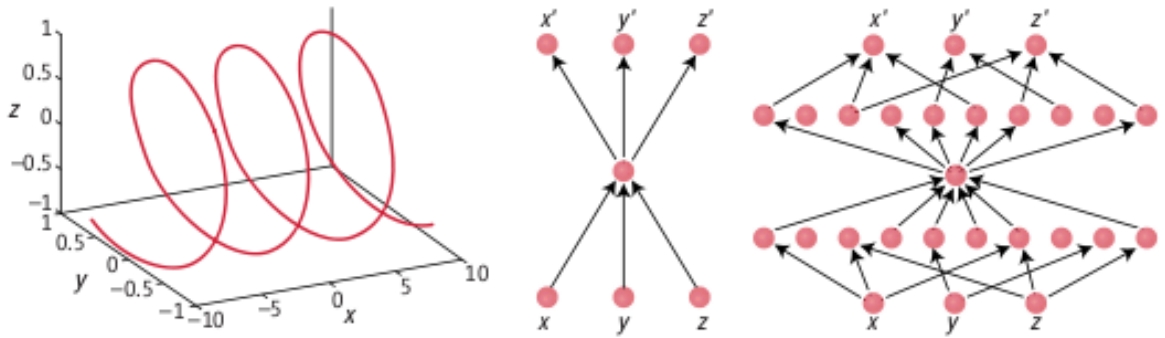
Relationship Between Approaches

Algorithm	Model	Encode	Decode	Relate Enc. & Dec.
PCA	Global Linear	✓	✓	$W_D = W_E^T$
ICA	Global Linear	✓	✓	$W_D = W_E^T$
Sparse Coding	Local Linear	✓($)	✓	$W_D = W_E^T$
PSD & LISTA	Local Linear	✓	✓	Learned W_E
DrLIM	Nonlinear	✓	X	Enc. Only
Auto-Encoders	Nonlinear	✓	✓	Learned W_E & W_D

	De-correlation/Independence
	Sparsity
	Metric Learning/Restricted Metric Learning
	All of the Above

Auto-Encoder Framework

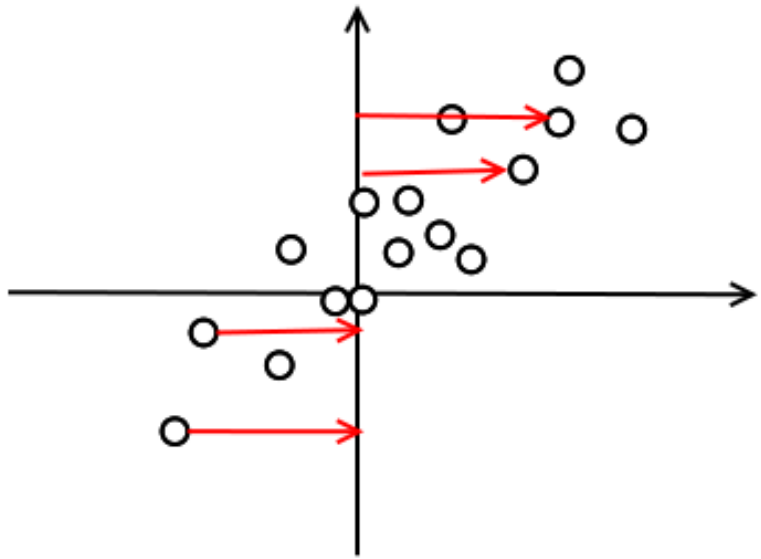
- An Auto-Encoder(AE) is composed of an "encoder" and "decoder"
- The encoder and decoder may correspond to completely different procedures (bases)
- The encoder transforms the data to latent space, and the decoder reconstructs the data from the latent representation
- AEs unify many data representation concepts and algorithms



Searching for structure. (**Left**) Three-dimensional data that are inherently one-dimensional. (**Middle**) A simple "autoencoder" network that is designed to compress three dimensions to one, through the narrow hidden layer of one unit. The inputs are labeled x, y, z, with outputs x', y', and z'. (**Right**) A more complex autoencoder network that can represent highly nonlinear mappings from three dimensions to one, and from one dimension back out to three dimensions.

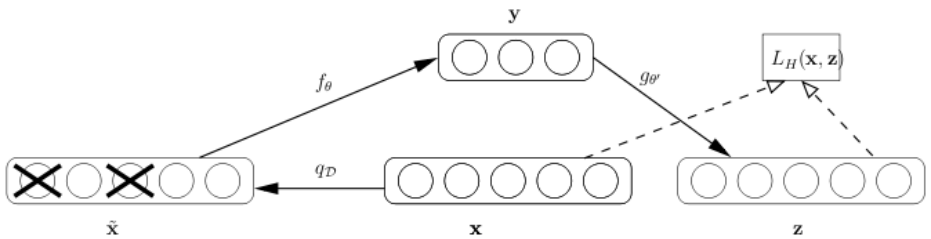
Linear AE and PCA

- Encoding and decoding simply corresponds to multiplication by W_e and W_d, respectively
- If *We* and *Wd* are full rank, then the AE has the capacity to learn the identity function
- Using s.g.d the largest eigenvectors of the covariance (or their mixtures) are learned first
- More interesting representations can be hoped for if some nonlinear function is applied to the encoder outputs
- AEs with nonlinearity can be stacked

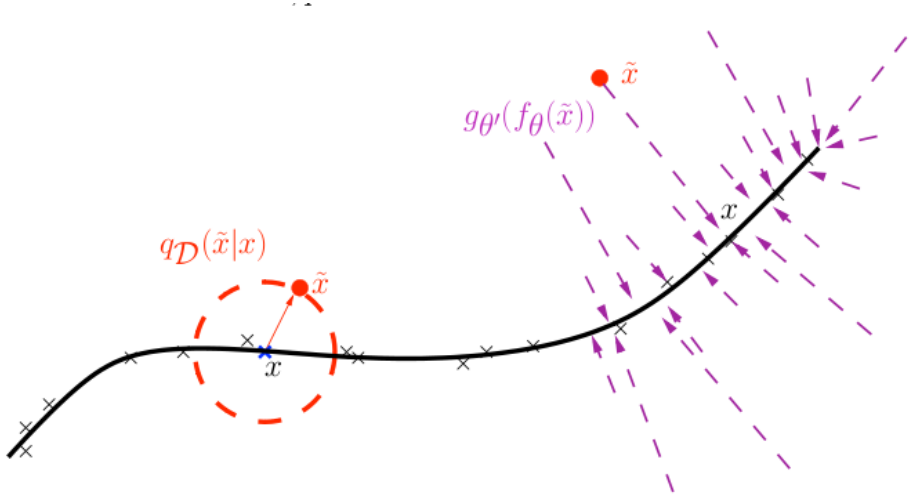


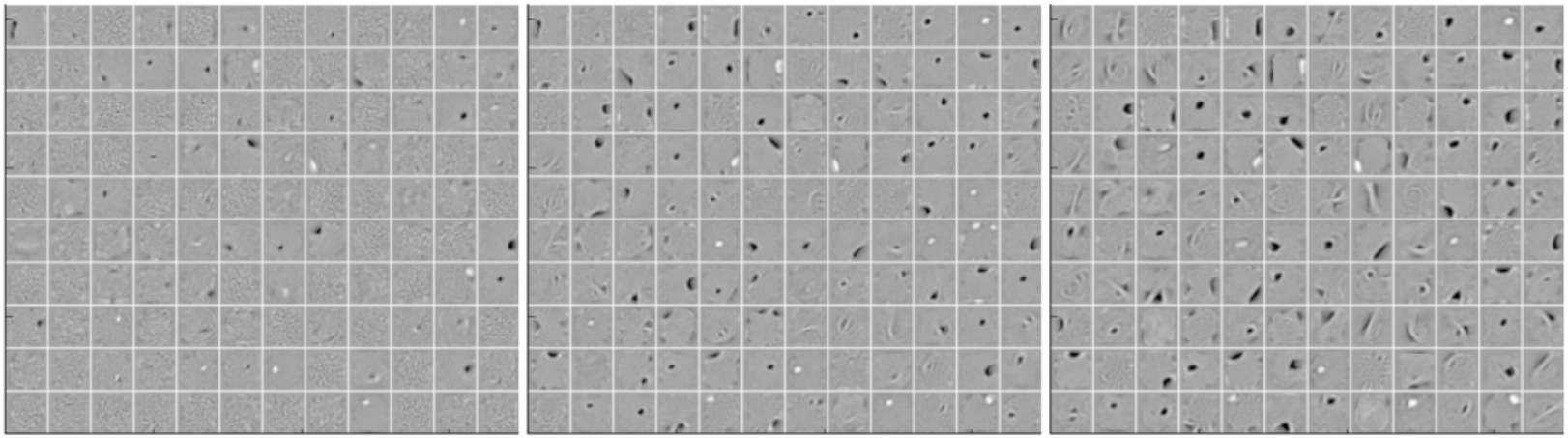
Denoising AE

- This AE is trained to reconstruct an image from a corrupted version of itself
- The corruption process randomly chooses some proportion of the pixels in the image and sets them to zero
- This explicitly forces the AE to learn and exploit dependencies between pixels in images
- Can be interpreted as simultaneously learning $p(x|\tilde{x})$ and how to infer x from $\tilde{x}$
- If $dim(Y) < dim(X)$ then "$Y = f(X)$ is a representation of X which is well suited for capturing the main variations in the data, i.e. on the manifold"



Denoising AE



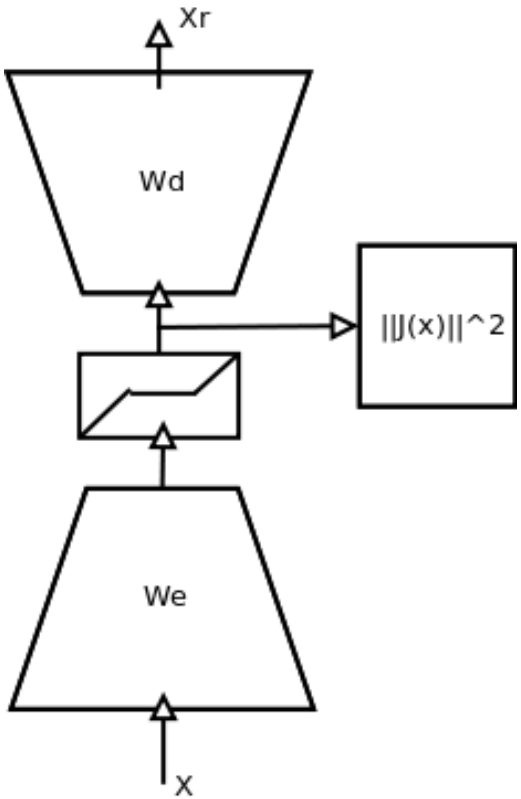


(a) No destroyed inputs (b) 25% destruction (c) 50% destruction

(d) Neuron A (0%, 10%, 20%, 50% destruction) (e) Neuron B (0%, 10%, 20%, 50% destruction)

Contractive AE

- Other techniques of obtaining "interesting" representations is to regularize the latent representation
- Contrary to more traditional regularization directly on weights, these correspond to "generic prior hypotheses"
- $L_{CAE} = \sum_{x \in D_n} \|x - x_r\|_2^2 + \lambda \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2$

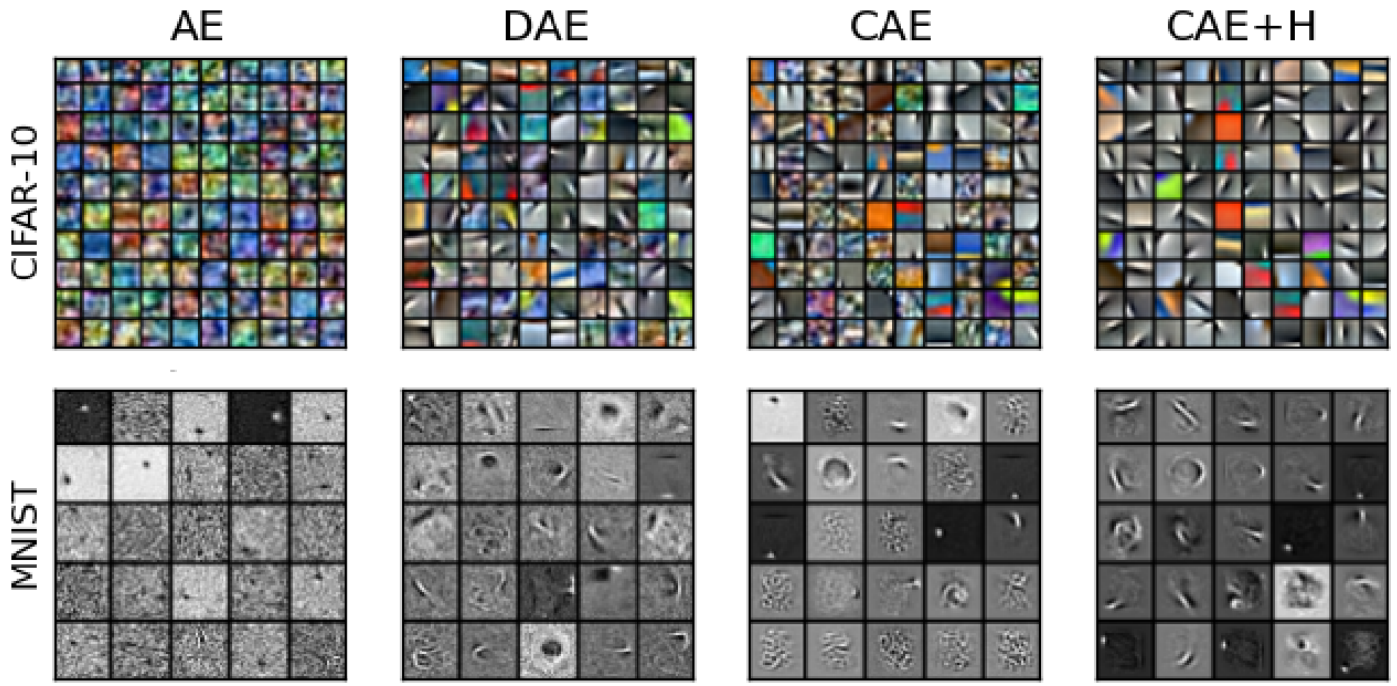


Contractive AE

- $\left(\frac{\partial h_j(x)}{\partial x_i}\right)^2 = (h'_j(x))^2 W^2_{e_{ij}}$
- Important to tie or normalize weights
- If there were no reconstruction objective then the penalty on the Jacobian would produce a constant representation for all inputs either by saturating or weight decay
- However nearby images on the manifold must be reconstructed as distinct images
- The contractive pressure is counteracted by the reconstruction gradient in the directions tangent to the manifold
- Contractive penalty has a component in the direction of curvature gradient
- Can be interpreted roughly as curvature regularization

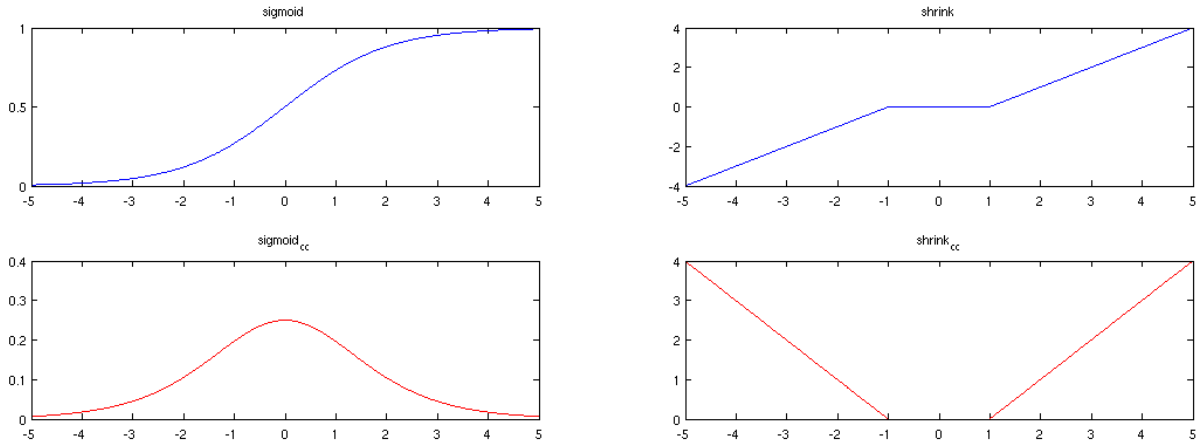
Relationship with other Auto-Encoders

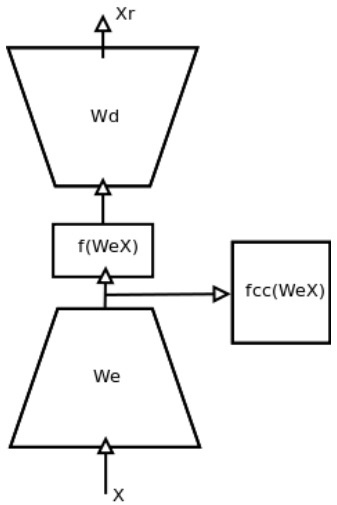
- Denoising auto-encoders also achieve a contractive mapping of the *reconstruction* via a stochastic process, which indirectly encourages invariance in the latent representation
- Sparse auto-encoders encourage the activations to be zero which occurs in the saturated part of the nonlinearity (assuming a proper choice of nonlinearity, e.g. LISTA)



Saturating Auto-Encoder (SAE)

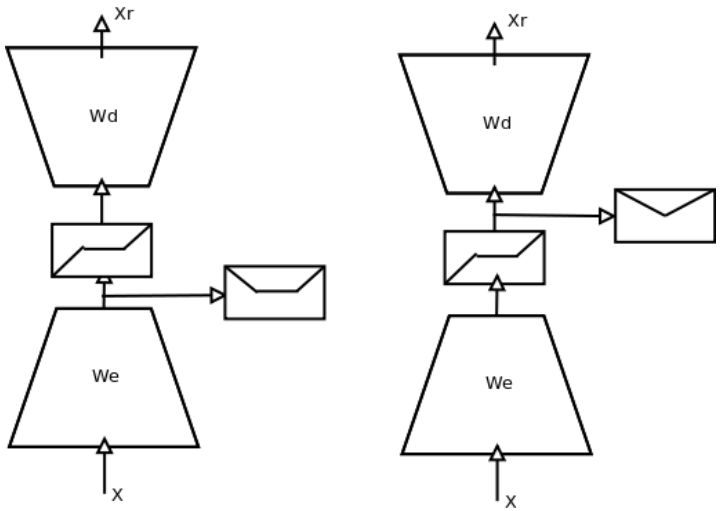
- Inspired by the CAE, we introduce a penalty on activations outside the saturated region of the nonlinearity

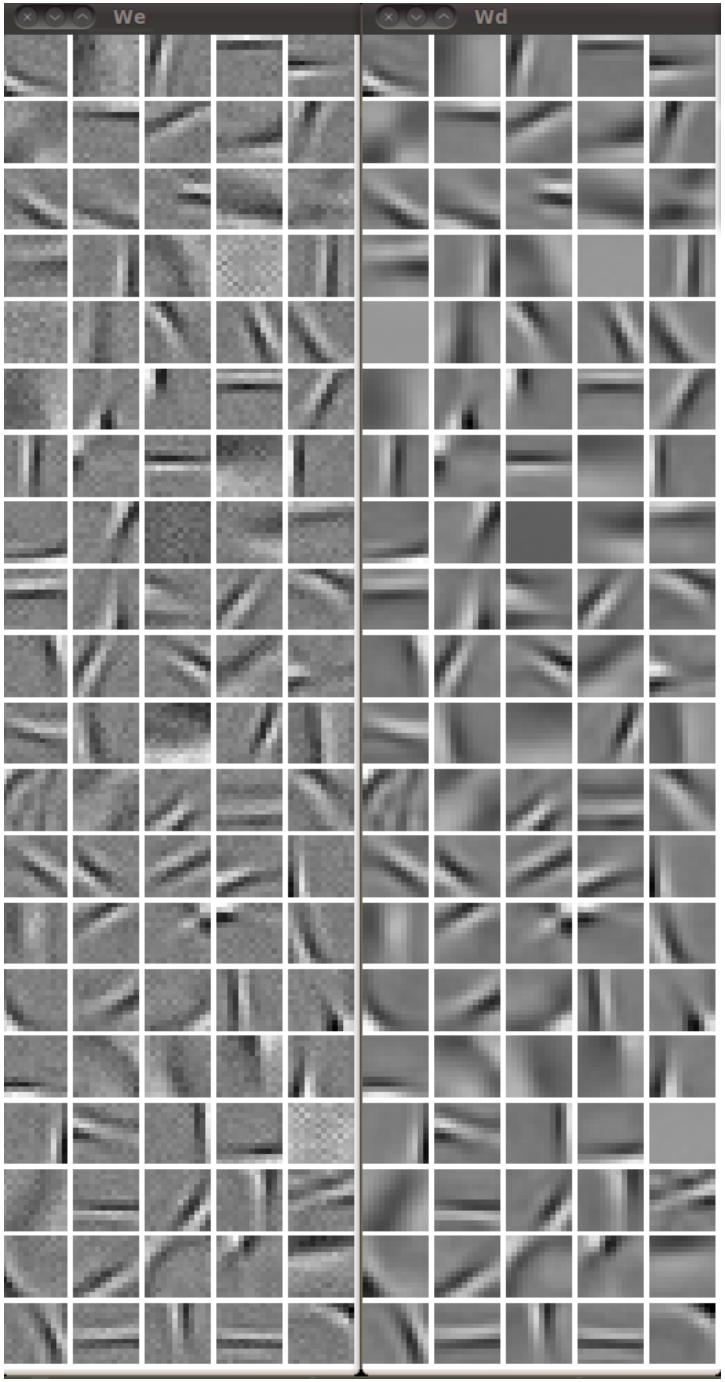




SAE with *shrink*() Nonlinearity

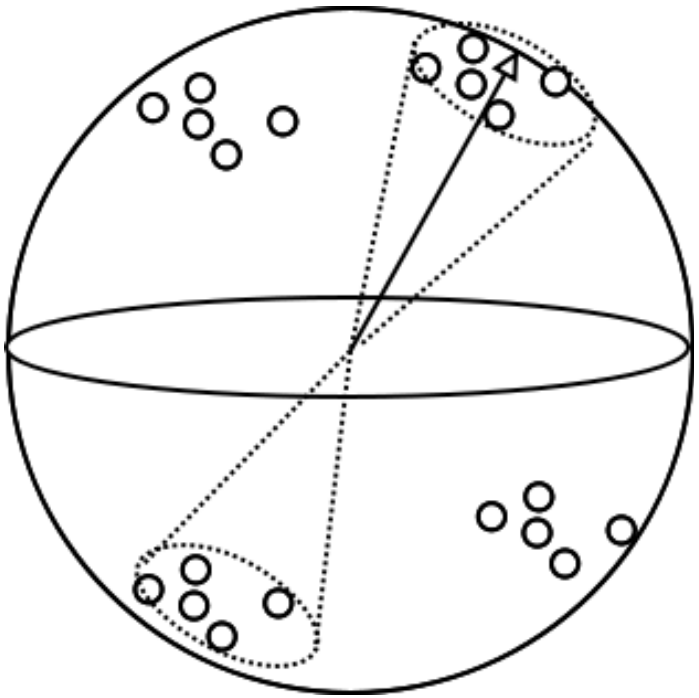
- The saturation penalty exactly corresponds to an L_1 penalty
- This auto-encoder strongly resembles sparse coding using PSD with *shrink*() nonlinearity
- Important to enforce $\|w_d\|_2 = 1$

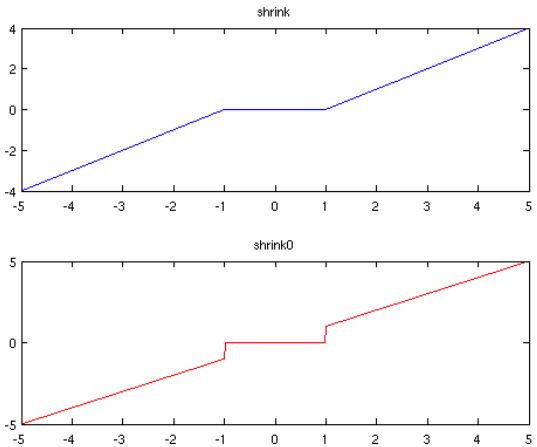



We
Wd

Geometric Interpretation

- Assume all the data points (circles) and decoder bases (arrow) are normalized to unity
- $\frac{\partial E_{rec}}{\partial W_d} = (x - x_{rec})shrink(W_e x)^T$
- The *shrink*() function explicitly promotes locality (specialization) of the basis elements by restricting the range of influence to nearby data points



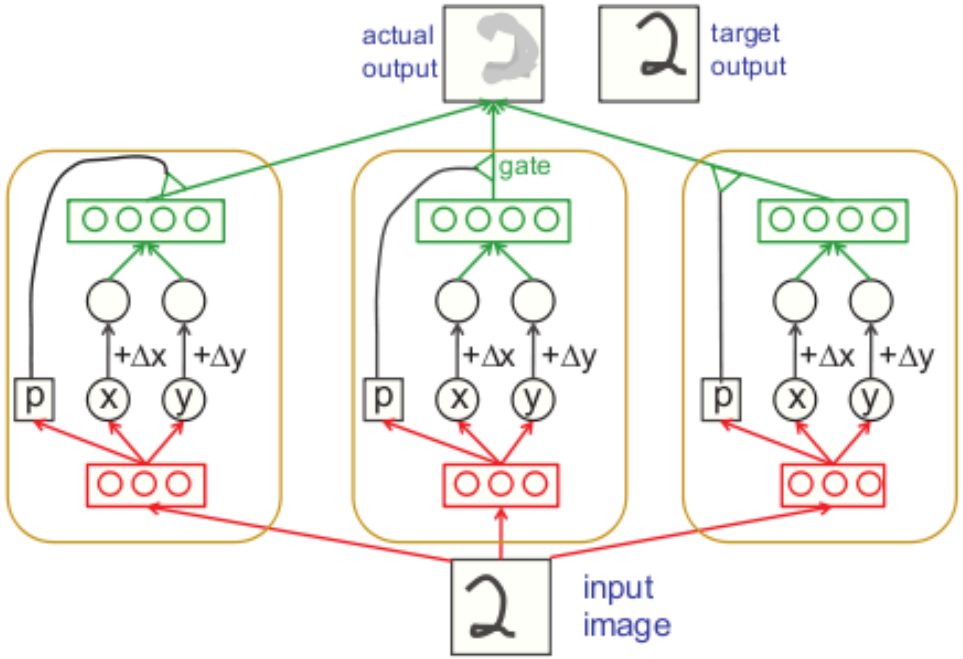


Interpretation of Nonlinearities

Enforcing a Factorization: Transforming Auto-Encoders

- "Move to a space in which the common variabilities can be described by linear transformations" (Hinton, 2012)
- In the case of objects in images: A representation in which the identity of the object (what?) is separately represented from the configuration (where?) would satisfy the above requirement
- This corresponds to a physically interpretable latent representation

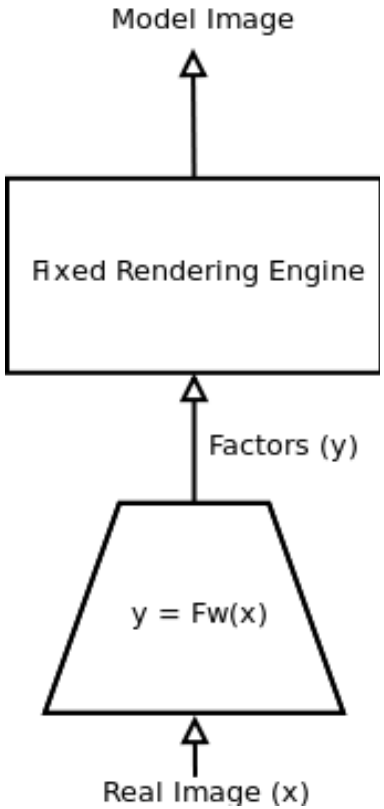
Enforcing a Factorization: Transforming Auto-Encoders



Each capsule has the capacity to represent only one instance of its visual entity at a time

Inverting Auto-Encoder

- $R_W(G(y_i')) = y_i''$ if $R_W = G^{-1}$ then $y_i' = y_i''$ thus we want to $min_W \|y_i' - y_i''\|$



Algorithm for training a recognition network R_w parameterized by weight vector w:

Given: Training set X of n data vectors $\{x_1, x_2, ..., x_n\}$, generative black box G, prototype code vector p.

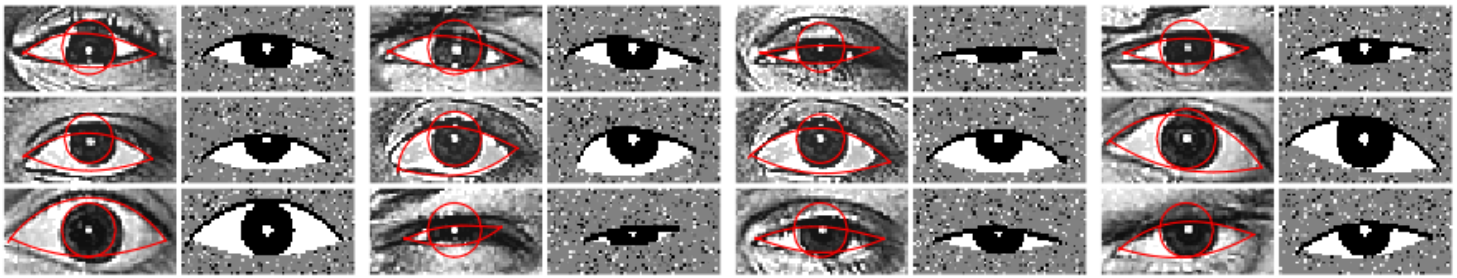
Initialization: Set output biases of R_w using p, and the remaining weights to samples from a zero-mean Gaussian with a small standard deviation.

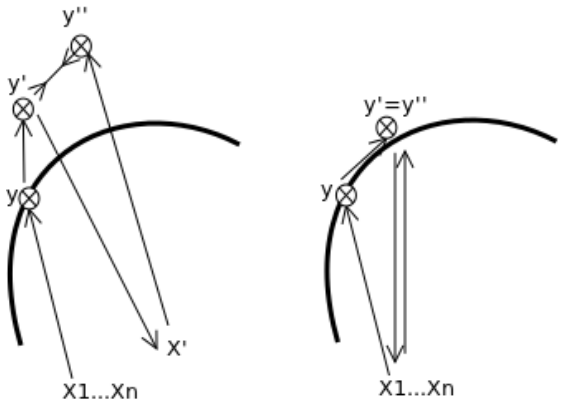
Weight update computed using the i^{th} (unlabelled) training case x_i:
Let y_i be the code vector inferred from x_i using the current recognition network R_w.

1. $y_i = R_w(x_i)$.
2. Perturb y_i randomly to create y_i'. (Note: The exact perturbation method is specified later.)
3. $x_i' = G(y_i')$.
4. Supervised learning on (x_i', y_i'):
   (a) $y_i'' = R_w(x_i')$.
   (b) $E = \|y_i' - y_i''\|^2$.
   (c) $w \leftarrow w - \eta \frac{\partial E}{\partial w}$.

Fig. 1. Summary of the breeder learning algorithm.

Inverting Auto-Encoder





Thank You

THE END