Representation of High Dimensional Data

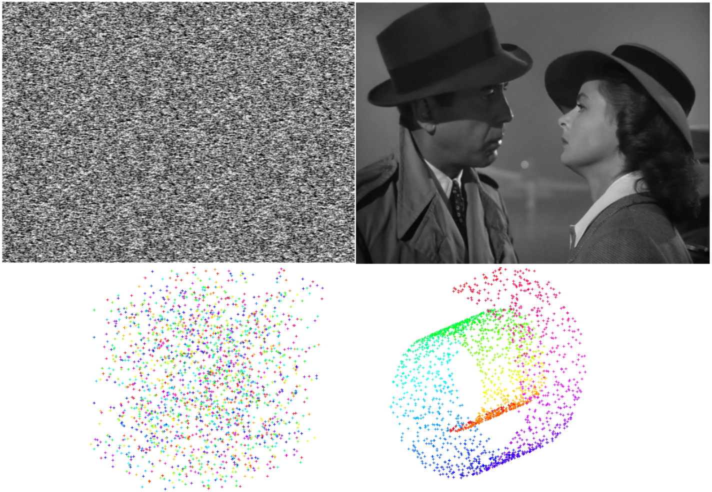
Ross Goroshin

July 9, 2012

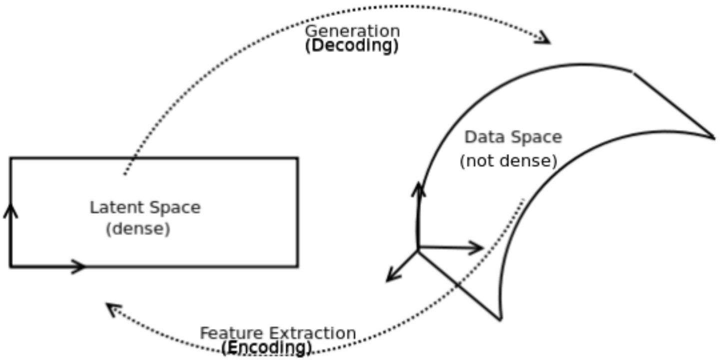
NEW YORK UNIVERSITY

Statistical Dependence and Natural Data Manifolds

From previous colloquium...



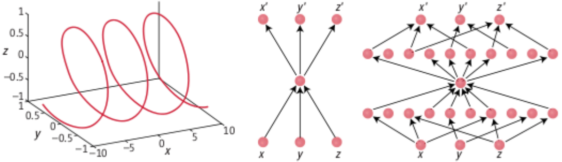
Dimensionality of Data & Dependence



- 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

Auto-Encoder Framework

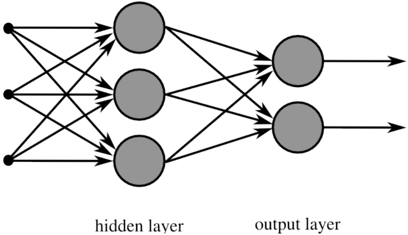
- An Auto-Encoder(AE) is composed of an "encoder" and "decoder" (matrices in the simplest case)
- The encoder and decoder may correspond to completely different procedures (bases)
- The encoder transforms the data to latent space, and the decoder reconstructs the data from the latent representation
-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



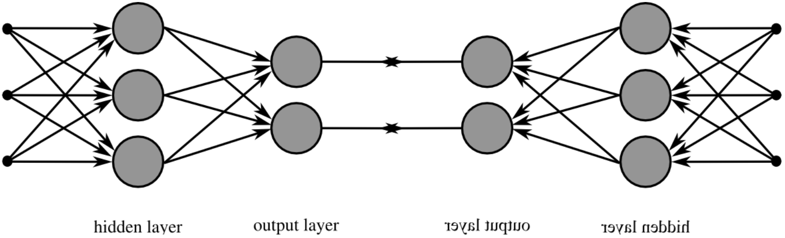
Searching for structure. **(Left)** Three-dimensional data that are inherently one-dimensional. **(Middle)** A simple "autoencoder" network that is designed to compress three dimensions to one, through the narrow hidden layer of one unit. The inputs are labeled *x*, *y*, *z*, with outputs *x*′, *y*′, 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.

Example: $x = cos(t), y = sin(t), z = t$

Neural Network



Auto-Encoder Network



Unsupervised and Deep Learning

- Auto-encoders have been instrumental in the development of "Deep Learning"
- "Deep Learning" based techniques leverage the inherently hierarchical nature of our world
- Reconstruction cost is often used for unlabeled data
- Overcomes the so called "vanishing gradient" problem in neural networks
- Allows for 'wholesale' propagation of information, i.e. labels provide only a few bits for information
- Connections with maximizing likelihood, except that the partition function is missing
- $p(x)$ usually tells you almost everything you want to know, e.g. $p(y|x)$ where y is some label

Simple Auto-Encoders

Let X be the input (image), W_e is the encoding weight matrix (basis), f is an arbitrary nonlinear function applied point-wise, and X_r is called the reconstruction.

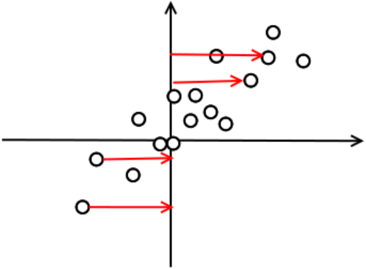
$$X_r = W_d f(W_e X + b_e) + b_d$$

$$min_{W_e, W_d, b_e, b_d} \sum_{X \in D} \frac{1}{2} \|X - X_r\|^2$$

- Interesting/useful representations are obtained when Auto-Encoders are forced learn to compress the data
- Informational bottleneck is established via W_e or f, or both
- Cost is minimized usually via *stochastic gradient descent*
- If f is identity and we enforce that $W_e = W_d^T$ then a nearly orthogonal basis is learned
- If $W_e \neq W_d^T$ then $W_d \approx W_e f^{-1}$ (...why?)

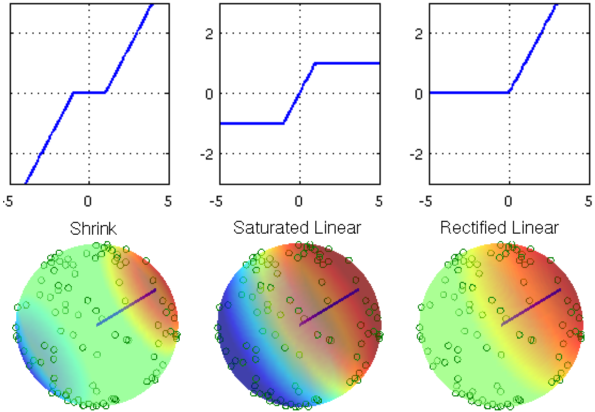
Linear AE and PCA

- 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



Visualizing the Role of f

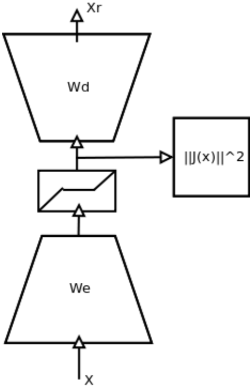
Assume that the data is normalized, i.e. lies on a sphere



$$\frac{\partial E}{\partial W_d} = (X - X_r) f(W_e X)^T$$
$$\frac{\partial E}{\partial W_e} = W_d (X - X_r) f'(W_e X)^T X$$

Information Bottleneck via Regularization: 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"
- $L_{CAE} = \sum_{x \in D_n} \|x - x_r\|_2^2 + \lambda \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2$

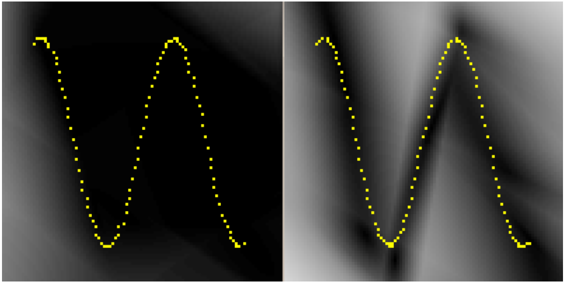


Contractive AE

- $\left(\frac{\partial h_j(x)}{\partial x_i}\right)^2 = (h'_j(x))^2 W_{e_{ij}}^2$
- 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

Visualizing the Effect of (Good) Regularization

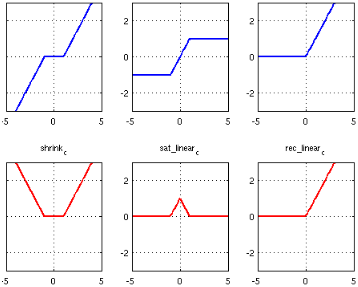
- The data manifold (distribution) is depicted as yellow points
- Brightness is proportional to reconstruction error (i.e. darker areas are better reconstructed)
- Implicitly parameterizes the data manifold
- Regularization is analogous to the partition function in max likelihood models



No Regularization   With Regularization

SATAE: Regularization via Saturation

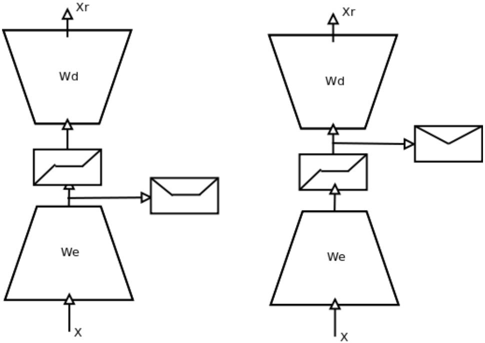
- Another regularization strategy is to encourage "activations" (h_i) in the saturation regime of the non-linearity (a.k.a activation function
- In all cases, this should bound the volume of space which is well reconstructed



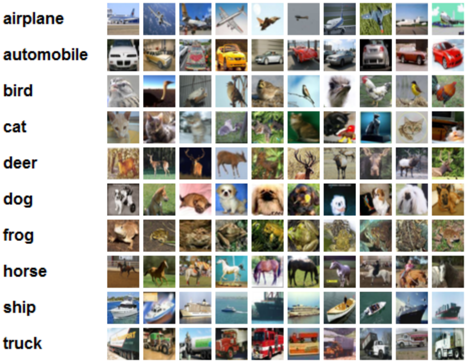
$$\text{Let } S = \{z \mid f'(z) = 0\}$$
$$f_c(z) = \inf_{z' \in S} |z - z'|$$

SATAE with *shrink*() Nonlinearity

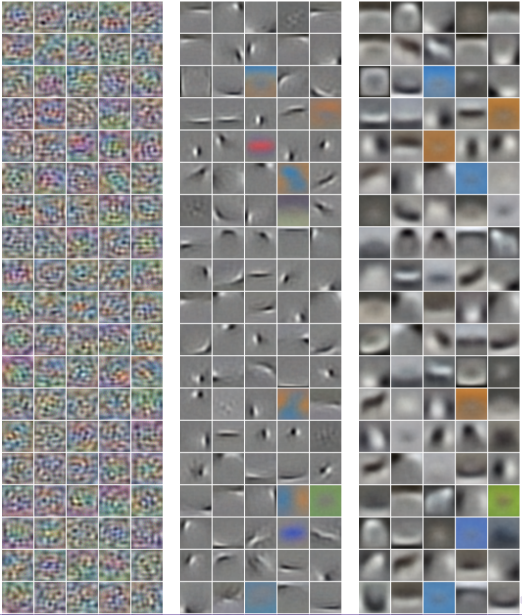
- The saturation penalty exactly corresponds to an L_1 penalty
- This auto-encoder strongly resembles sparse coding using PSD with *shrink*() nonlinearity
- Important to enforce $\|w_d\|_2 = 1$



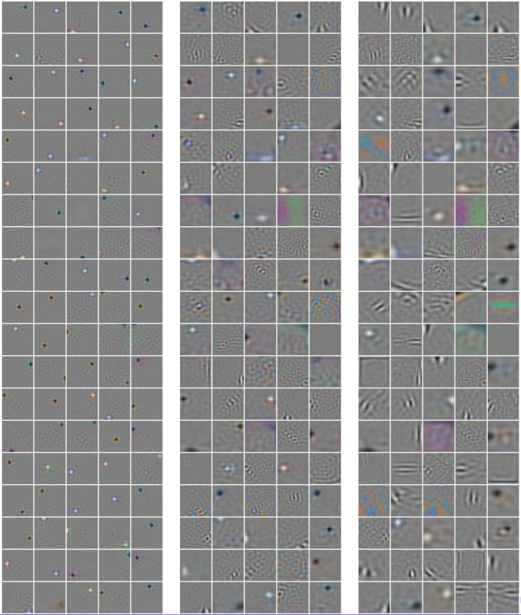
Experiments on Images



SATAE-shrink



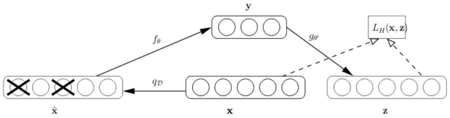
SATAE-saturated linear



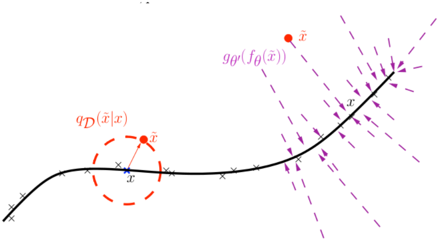
SATAE-saturated linear Progressively Increasing Saturation

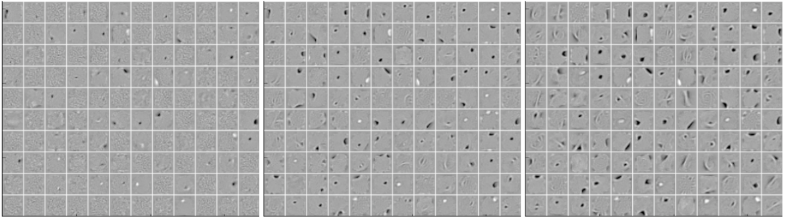
Denoising AE

- 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"

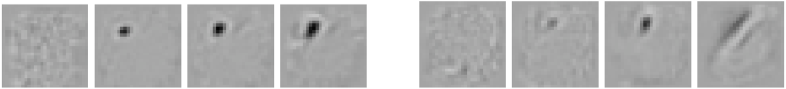


Denoising AE





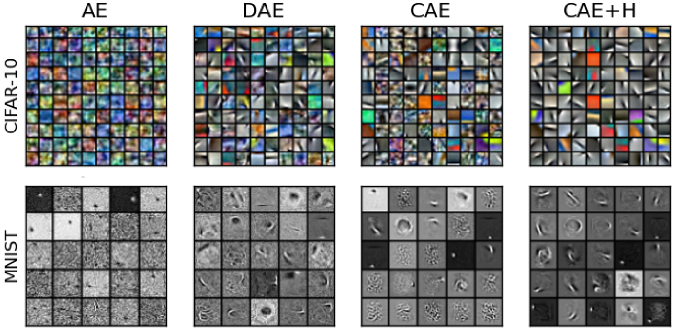
(a) No destroyed inputs (b) 25% destruction (c) 50% destruction



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

Relationship with other Auto-Encoders

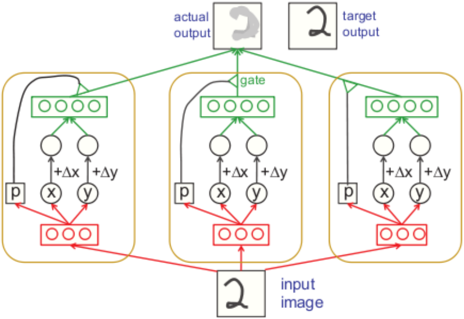
- Denoising auto-encoders also achieve a contractive mapping of the *reconstruction* via a stochastic process, which indirectly encourages invariance in the latent representation
- Sparse auto-encoders encourage the activations to be zero which occurs in the saturated part of the nonlinearity (assuming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



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

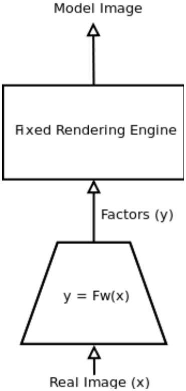
Enforcing a Factorization: Transforming Auto-Encoders



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.

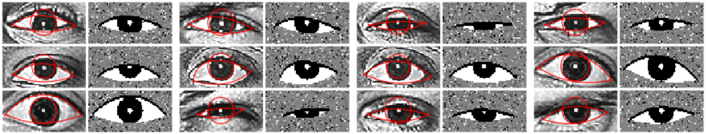
Initialization: Set output biases of R_w using p, and the remaining weights to samples from a zero-mean Gaussian with a small standard deviation.

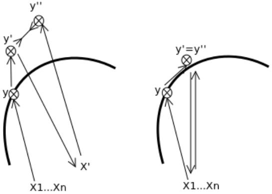
Weight update computed using the i^{th} (unlabelled) training case x_i:
Let y_i be the code vector inferred from x_i using the current recognition network R_w.

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