Autoencoders

Jiahui Chen
Department of Mathematical Sciences
University of Arkansas

Reference: Hinton's notes, https://www.edureka.co/blog/autoencoders-tutorial

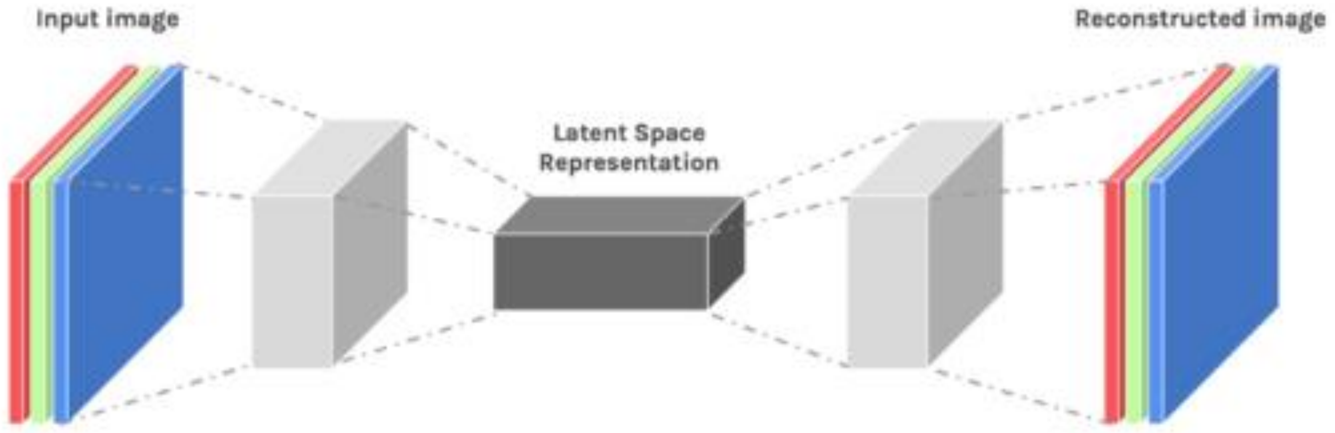
UNIVERSITY OF ARKANSAS

Autoencoders

- Supervised learning uses explicit labels/correct output in order to train a network.
  - E.g., classification of images.
- Unsupervised learning relies on data only.
  - Output is determined implicitly from word order in the input data (Word2Vec, CBOW and the Skip-gram models in NLP).
  - Key point is to produce a useful embedding of words.
  - The embedding encodes structure such as word similarity and some relationships.
  - Still need to define a loss – this is an implicit supervision.

Autoencoders

- Autoencoders are designed to reproduce their input, especially for images.
  - Key point is to reproduce the input from a learned encoding.



Autoencoders

- Compare PCA/SVD
  - PCA takes a collection of vectors (images) and produces a usually smaller set of vectors that can be used to approximate the input vectors via linear combination.
  - Very efficient for certain applications.
  - Fourier and wavelet compression is similar.

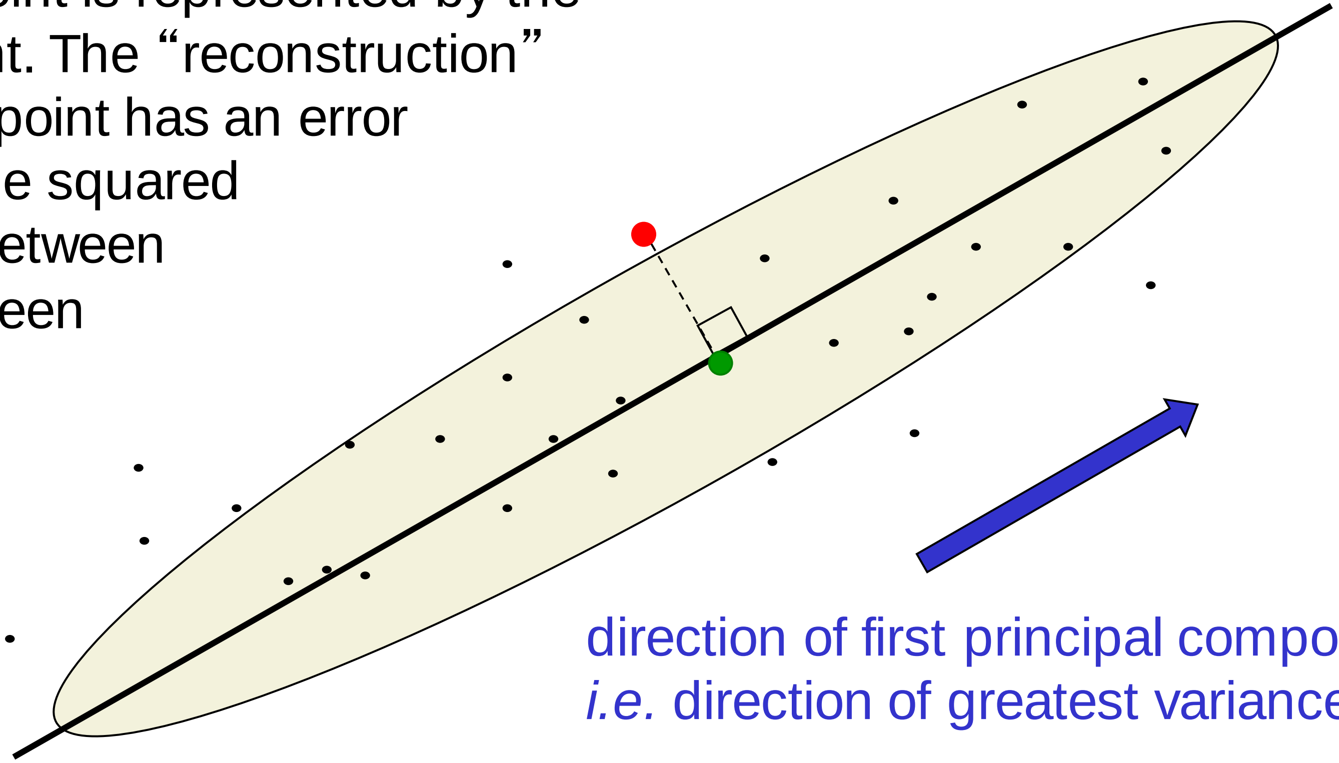
- Neural network autoencoders
  - Can learn nonlinear dependencies
  - Can use convolutional layers
  - Can use transfer learning

Principal Components Analysis

- This takes N-dimensional data and finds the M orthogonal directions in which the data have the most variance.
  - These M principal directions form a lower-dimensional subspace.
  - We can represent an N-dimensional datapoint by its projections onto the M principal directions.
  - This loses all information about where the datapoint is located in the remaining orthogonal directions.
- We reconstruct by using the mean value (over all the data) on the N-M directions that are not represented.
  - The reconstruction error is the sum over all these unrepresented directions of the squared differences of the datapoint from the mean.

PCA

The red point is represented by the green point. The “reconstruction” of the red point has an error equal to the squared distance between red and green points.

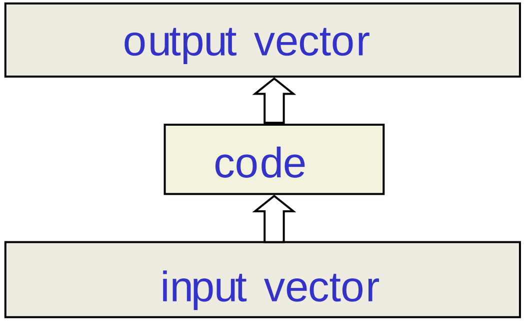


direction of first principal component

i.e. direction of greatest variance

Backpropagation with PCA

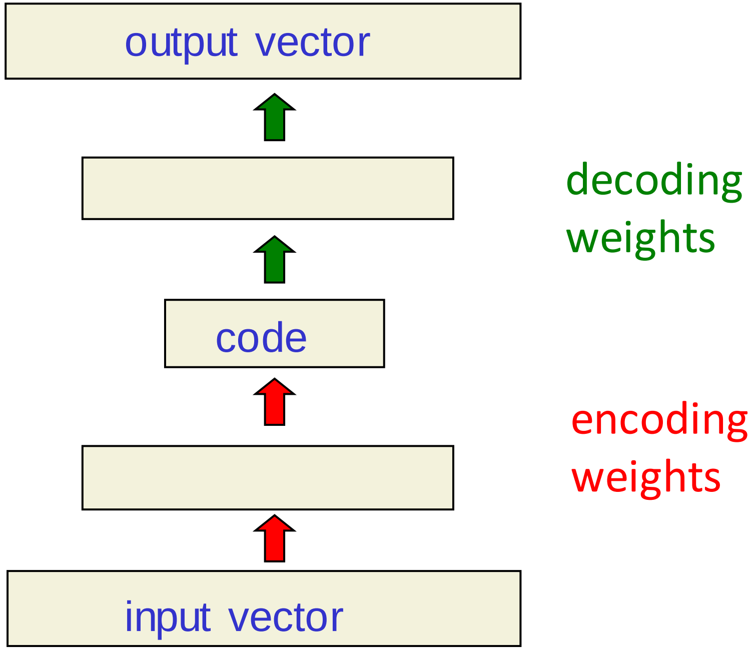
- Try to make the output be the same as the input in a network with a central bottleneck.



- The activities of the hidden units in the bottleneck form an efficient code.
- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared reconstruction error.
  – This is exactly what PCA does.
- The M hidden units will span the same space as the first M components found by PCA
  – Their weight vectors may not be orthogonal.
  – They will tend to have equal variances.

Using Backpropagation to Generalize PCA

- With non-linear layers before and after the code, it should be possible to efficiently represent data that lies on or near a non-linear manifold.
  - The encoder converts coordinates in the input space to coordinates on the manifold.
  - The decoder does the inverse mapping.

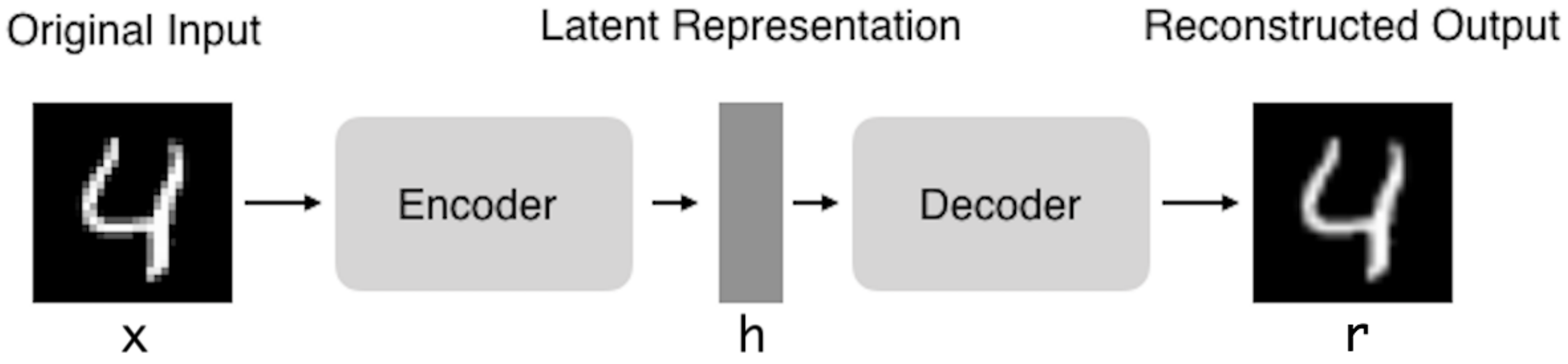


Autoencoders

- They always looked like a really nice way to do non-linear dimensionality reduction:
  - They provide flexible mappings both ways.
  - The learning time is linear (or better) in the number of training cases.
  - The final encoding model is fairly compact and fast.
- But it turned out to be very difficult to optimize deep autoencoders using backpropagation.
  - With small initial weights the backpropagated gradient dies.
- We now have a much better ways to optimize them.
  - Use unsupervised layer-by-layer pre-training.
  - Or just initialize the weights carefully as in Echo-State Nets.

Autoencoders: Structure

- Encoder: compress input into a latent-space of usually smaller dimension. h = f(x)
- Decoder: reconstruct input from the latent space. r = g(f(x)) with r as close to x as possible



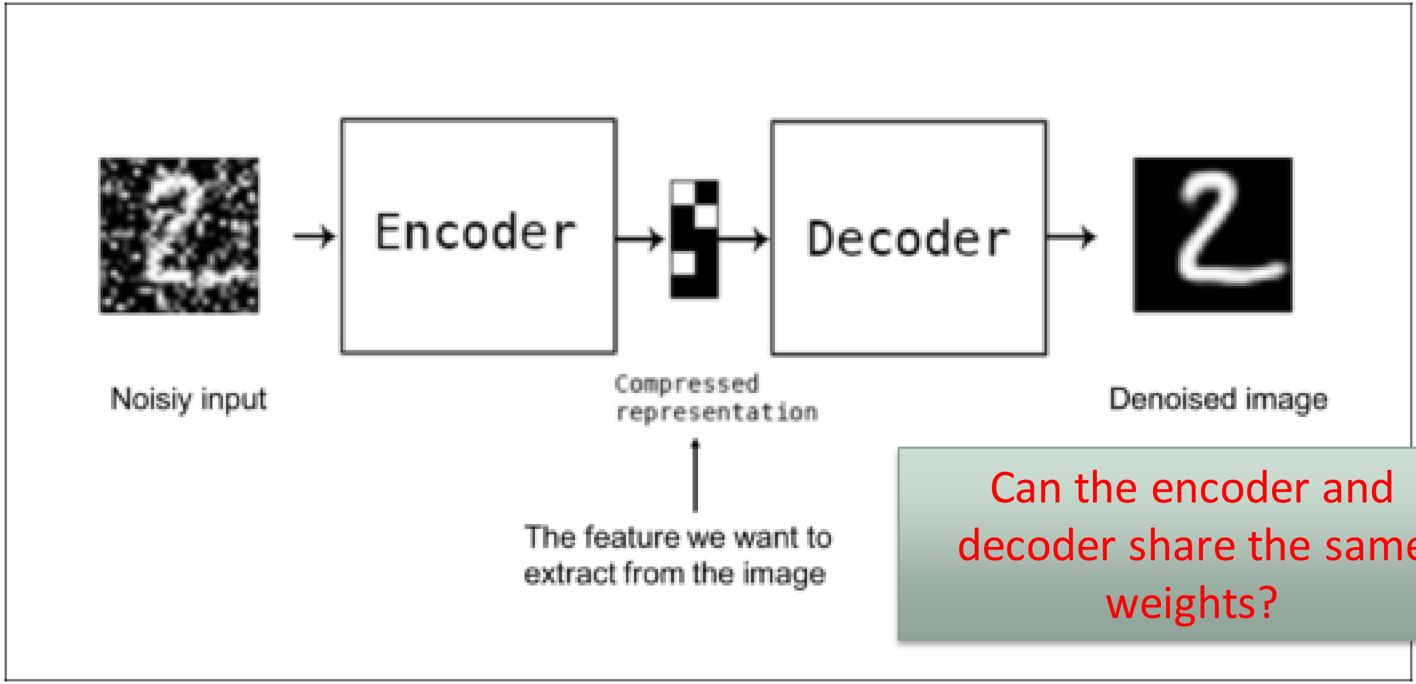
Autoencoders: Structure

- The first really successful deep autoencoders (Hinton & Salakhutdinov, Science, 2006)

$$\begin{tikzcd}
784 \arrow[r, "W_1"] & 1000 \arrow[r, "W_2"] & 500 \arrow[r, "W_3"] & 250 \arrow[dr, "W_4"] & \\
 & & & & \text{30 linear units} \arrow[dl, "W_4^T"] \\
784 & 1000 \arrow[l, "W_1^T"'] & 500 \arrow[l, "W_2^T"'] & 250 \arrow[l, "W_3^T"'] &
\end{tikzcd}$$

Autoencoders: Applications

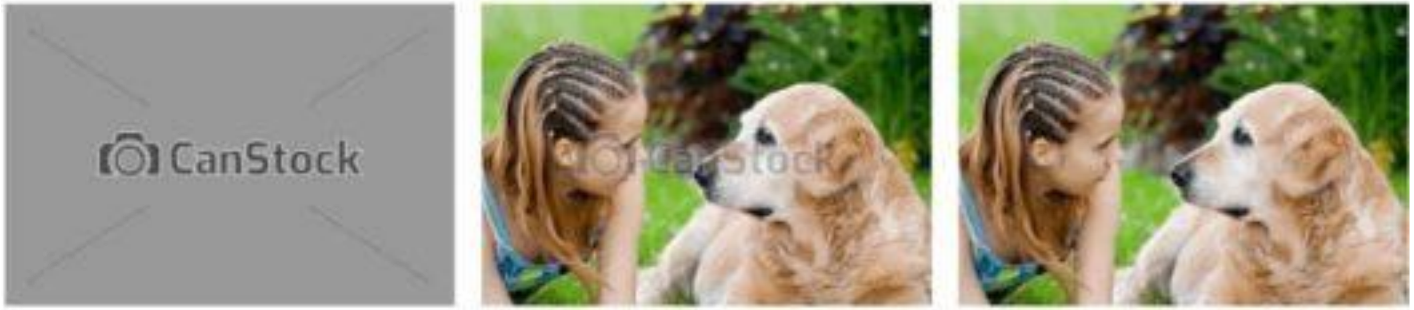
- Denoising: input clean image + noise and train to reproduce the clean image.


Encoder
Decoder
Noisiy input
Compressed representation
Denoised image
The feature we want to extract from the image

Can the encoder and decoder share the same weights?

Autoencoders: Applications

- Watermark removal



Autoencoders: Applications

- Image colorization: input black and white and train to produce color images

Properties of Autoencoders

- **Data-specific**: Autoencoders are only able to compress data similar to what they have been trained on.
- **Lossy:** The decompressed outputs will be degraded compared to the original inputs.
- **Learned automatically from examples:** It is easy to train specialized instances of the algorithm that will perform well on a specific type of input.

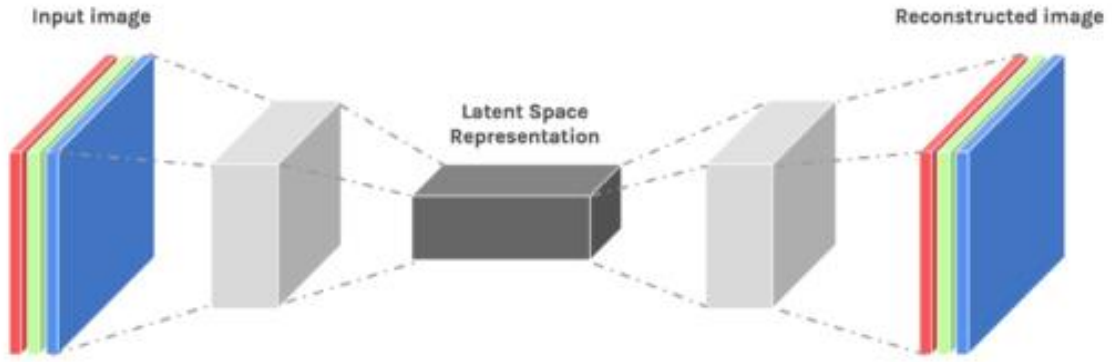
Properties of Autoencoders

- As with other NNs, overfitting is a problem when capacity is too large for the data.

- Autoencoders address this through some combination of:
  - Bottleneck layer – fewer degrees of freedom than in possible outputs.
  - Training to denoise.
  - Sparsity through regularization.
  - Contractive penalty.

Latent Space (Bottleneck)

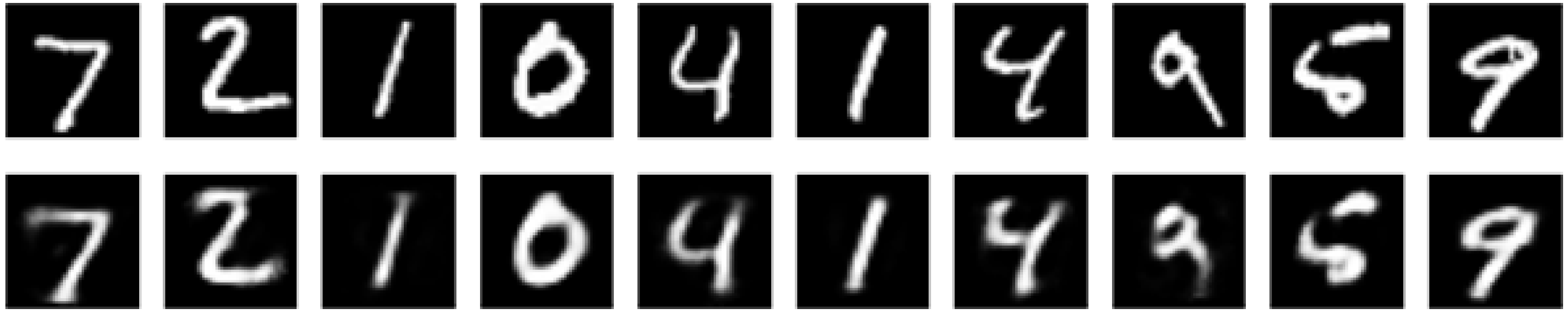
- Suppose input images are nxn and the latent space is m < nxn.
- Then the latent space is not sufficient to reproduce all images.
- Needs to learn an encoding that captures the important features in training data, sufficient for approximate reconstruction.



Latent Space in Keras

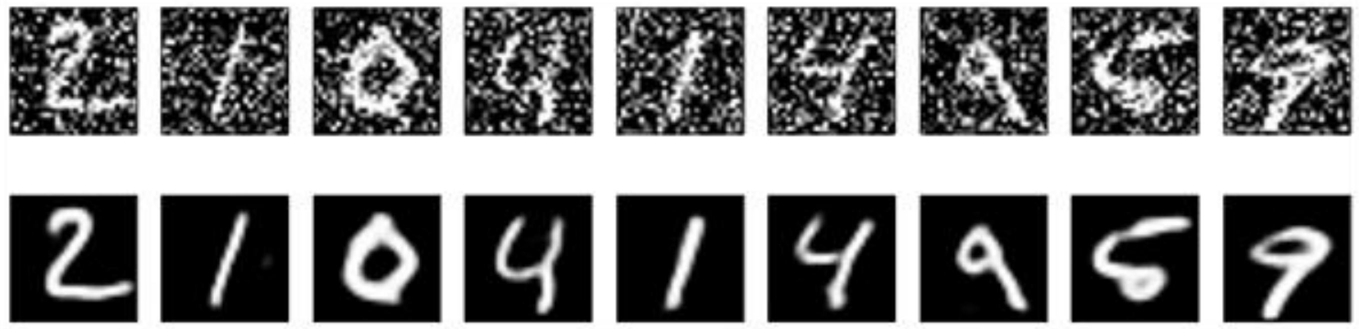
- input_img = Input(shape=(784,))
- encoding_dim = 32
- encoded = Dense(encoding_dim, activation='relu')(input_img)
- decoded = Dense(784, activation='sigmoid')(encoded)
- autoencoder = Model(input_img, decoded)

- Maps 28x28 images into a 32 dimensional vector.
- Can also use more layers and/or convolutions.

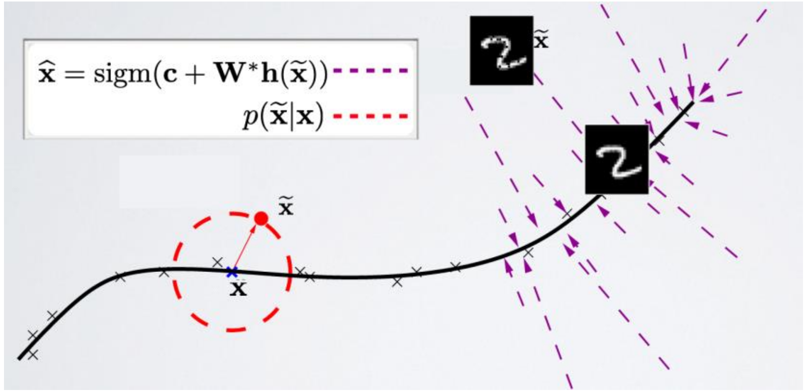


Denoising Autoencoders

- Basic autoencoder trains to minimize the loss between x and the reconstruction g(f(x)).
- Denoising autoencoders train to minimize the loss between x and g(f(x+w)), where w is random noise.
- Same possible architectures, different training data.
- Kaggle has a dataset on damaged documents.



Denoising Autoencoders



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

- Denoising autoencoders can't simply memorize the input output relationship.
- Intuitively, a denoising autoencoder learns a projection from a neighborhood of our training data back onto the training data.

Sparse autoencoders

- Construct a loss function to penalize *activations* within a layer.
- Usually regularize the *weights* of a network, not the activations.
- Individual nodes of a trained model that activate are *data-dependent.*
  - Different inputs will result in activations of different nodes through the network.
- Selectively activate regions of the network depending on the input data.

https://www.jeremyjordan.me/autoencoders/

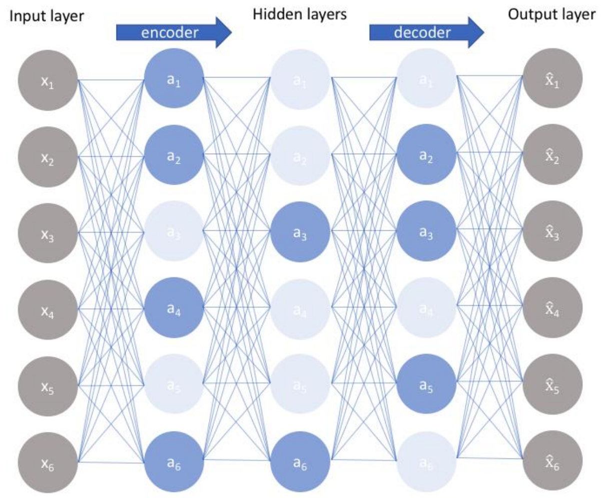
Sparse autoencoders

- Construct a loss function to penalize *activations* the network.
  - **L1 Regularization**: Penalize the absolute value of the vector of activations *a* in layer *h* for observation *I*

$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i \left| a_i^{(h)} \right|$$

  - **KL divergence:** Use cross-entropy between average activation and desired activation

$$\mathcal{L}(x, \hat{x}) + \sum_j KL(\rho || \hat{\rho}_j)$$

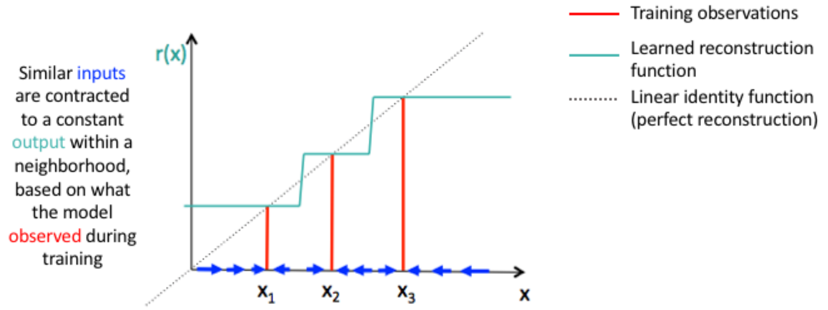


https://www.jeremyjordan.me/autoencoders/

Contractive autoencoders

- Arrange for similar inputs to have similar activations.
  - I.e., the *derivative of the hidden layer activations are small* with respect to the input.
- Denoising autoencoders make the *reconstruction function* (encoder+decoder) resist small perturbations of the input
- Contractive autoencoders make the *feature extraction function* (ie. encoder) resist infinitesimal perturbations of the input.

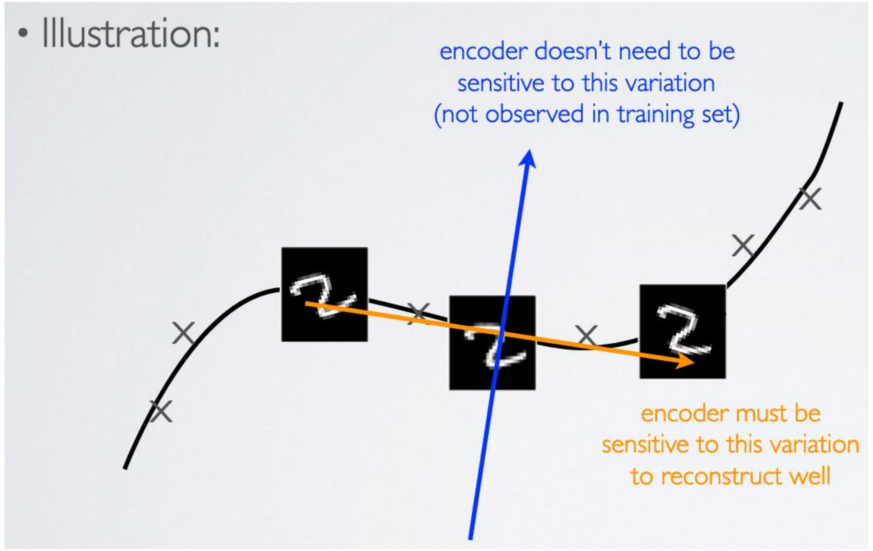
$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i \left\| \nabla_x a_i^{(h)}(x) \right\|^2$$


Similar inputs are contracted to a constant output within a neighborhood, based on what the model observed during training
r(x)
x_1
x_2
x_3
x
Training observations
Learned reconstruction function
Linear identity function (perfect reconstruction)

https://www.jeremyjordan.me/autoencoders/

Contractive autoencoders

- Contractive autoencoders make the *feature extraction function* (ie. encoder) resist infinitesimal perturbations of the input.


• Illustration:
encoder doesn't need to be sensitive to this variation (not observed in training set)
encoder must be sensitive to this variation to reconstruct well

https://ift6266h17.files.wordpress.com/2017/03/14_autoencoders.pdf

Autoencoders

- Both the denoising and contractive autoencoder can perform well
  - Advantage of denoising autoencoder : simpler to implement- requires adding one or two lines of code to regular autoencoder- no need to compute Jacobian of hidden layer
  - Advantage of contractive autoencoder : gradient is deterministic - can use second order optimizers (conjugate gradient, LBFGS, etc.)- might be more stable than denoising autoencoder, which uses a sampled gradient
- To learn more on contractive autoencoders:
  - Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot et Yoshua Bengio, 2011.

https://ift6266h17.files.wordpress.com/2017/03/14_autoencoders.pdf