Autoencoders

Vahid Tarokh

ECE 685D, Fall 2025

Unsupervised Learning

- Unsupervised learning: we only use the inputs $\mathbf{x}^{(t)}$ for learning
  - automatically extract meaningful features for your data
  - leverage the availability of unlabeled data
  - add a data-dependent regularizer to training (in some cases)

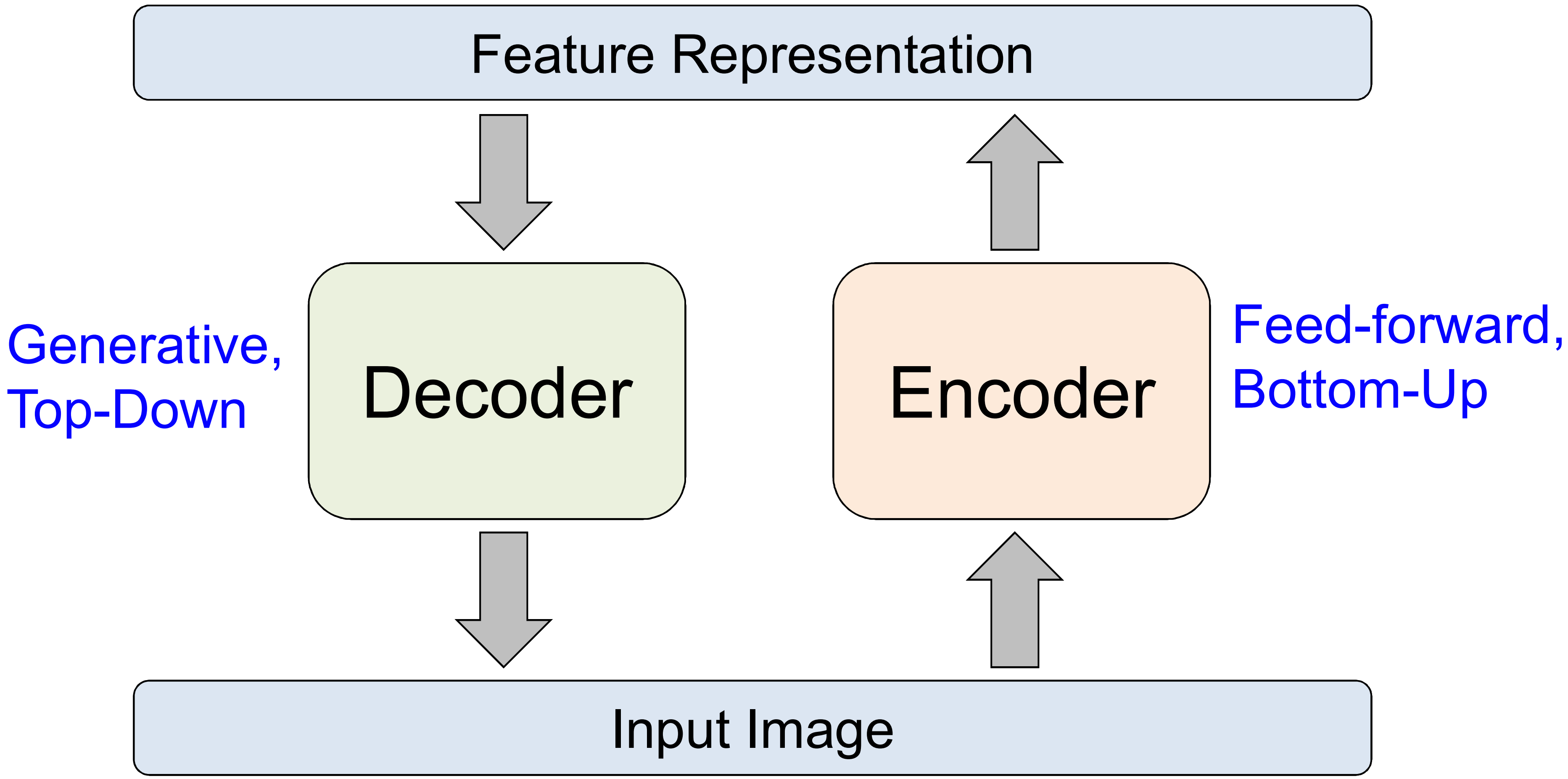
- We will consider 3 models for unsupervised learning that will form the basic building blocks for deeper models:
  - **Autoencoders**
  - Sparse Coding
  - Restricted Boltzmann Machines

Autoencoders

- Map high-dimensional data to lower dimensions
  - Visualization
  - Compression (reducing the file size)
- Learn abstract features in an unsupervised way for downstream supervised tasks
  - Unlabeled data usually are much more plentiful than labeled data
- Build some generative models

Autoencoders

An autoencoder is a feed-forward neural net whose job is to take an input x and output $\hat{x}$



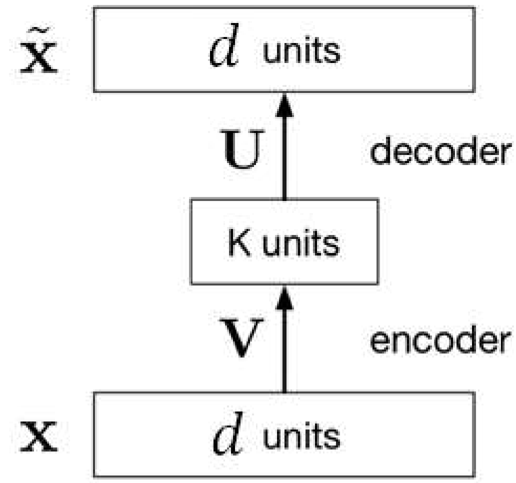
- Details of what goes insider the encoder and decoder matter!

Totally Linear Case: PCA

- The simplest kind of autoencoder has one hidden layer, linear activations, and squared error loss.

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$

- This network computes $\tilde{\mathbf{x}} = \mathbf{UVx}$, which is a linear function.
- If $K \geq d$, we can choose $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{UV}$ is the identity. This isn't very interesting.



If $k < d$, then we have dimensionality reduction.

A linear autoencoder with MSE loss, one hidden layer, and tied weights learns the exact same subspace as PCA. Autoencoders are a nonlinear generalization of PCA.

PCA

- Dimensionality reduction technique for data in $\mathbb{R}^d$
- Problem statement
  - given m points $x_1, \ldots, x_m$ in $\mathbb{R}^d$
  - and target dimension $k < d$
  - find "best" k-dimensional subspace approximating the data
- Formally: find matrices $U \in \mathbb{R}^{d \times k}$ and $V \in \mathbb{R}^{k \times d}$
- that minimize

$$f(U, V) = \sum_{i=1}^{m} \|x_i - UVx_i\|_2^2$$

- $V : \mathbb{R}^d \to \mathbb{R}^k$ is "compressor", $U : \mathbb{R}^k \to \mathbb{R}^d$ is "decompressor"
- Is $f : \mathbb{R}^{d \times k} \times \mathbb{R}^{k \times d} \to \mathbb{R}$ convex?
- No! $f(U, \cdot)$ and $f(\cdot, V)$ both convex, but not $f(\cdot, \cdot)$
- **Claim:** optimal solution achieved at $U = V^\top$ and $U^\top U = I$ (columns of U are orthonormal)

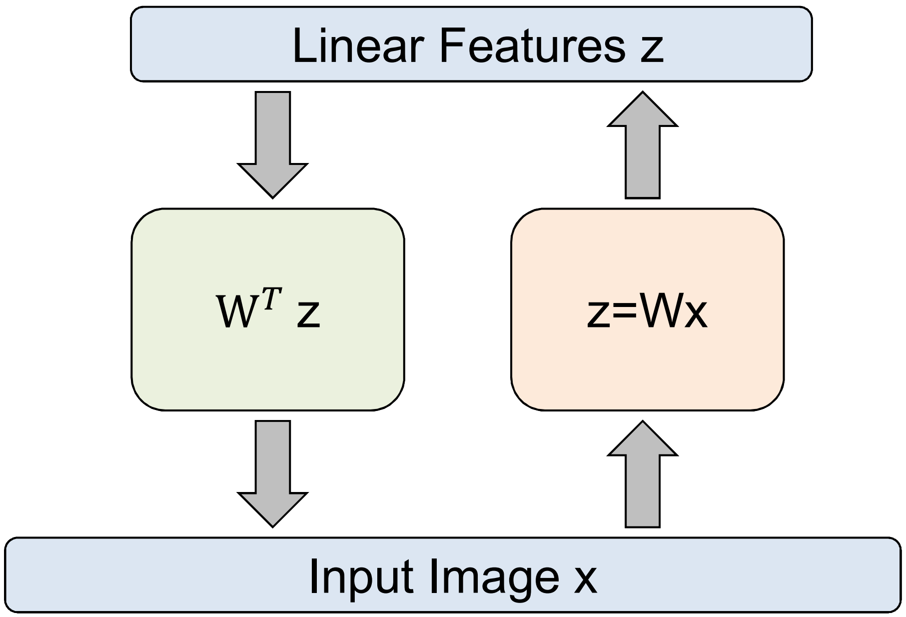
PCA

- Optimization problem: $\text{minimize}_{U \in \mathbb{R}^{d\times k}, V \in \mathbb{R}^{k \times d}} \sum_{i=1}^m \|x_i - UVx_i\|_2^2$
- **Claim:** optimal solution achieved at $U = V^\top$ and $U^\top U = I$
- Proof:
    - For any U, V, linear map $x \mapsto UVx$ has range R of dimension k
    - Let $w_1, \ldots, w_k$ be orthonormal basis for R; arrange into columns of W.
    - Hence, for each x_i there is $z_i \in \mathbb{R}^k$ such that $UVx_i = Wz_i$.
    - Note: $W^\top W = I$.
    - Which z minimizes $f(x_i, z) := \|x_i - Wz\|_2^2$?
    - For all $x \in \mathbb{R}^d, z \in \mathbb{R}^k$,
      $f(x, z) = \|x\|_2^2 + z^\top W^\top W z - 2z^\top W^\top x = \|x\|_2^2 + \|z\|_2^2 - 2z^\top W^\top x$.
    - Minimize w.r.t. z: $\nabla_z f = 2z - 2W^\top x = 0 \implies z = W^\top x$.
    - Therefore

$$\sum_{i=1}^m \|x_i - UVx_i\|_2^2 = \sum_{i=1}^m \|x_i - Wz_i\|_2^2 \geq \sum_{i=1}^m \|x_i - WW^\top x_i\|_2^2.$$

    - U, V are optimal, so $\sum_{i=1}^m \|x_i - UVx_i\|_2^2 = \sum_{i=1}^m \|x_i - WW^\top x_i\|_2^2$.
    - So instead of U, V can take $W, W^\top$. $\square$
- $WW^\top x$ is the **orthogonal projection** of x onto R.

Linear Autoencoder (PCA)



- 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 error.
- The K hidden units will span the same space as the first k principal components. The weight vectors may not be orthogonal.
- If $K < d$, and W with orthogonal $rows$, then we have PCA.

With nonlinear hidden units, we have a **nonlinear generalization of PCA.**

Optimality of the Linear Autoencoder

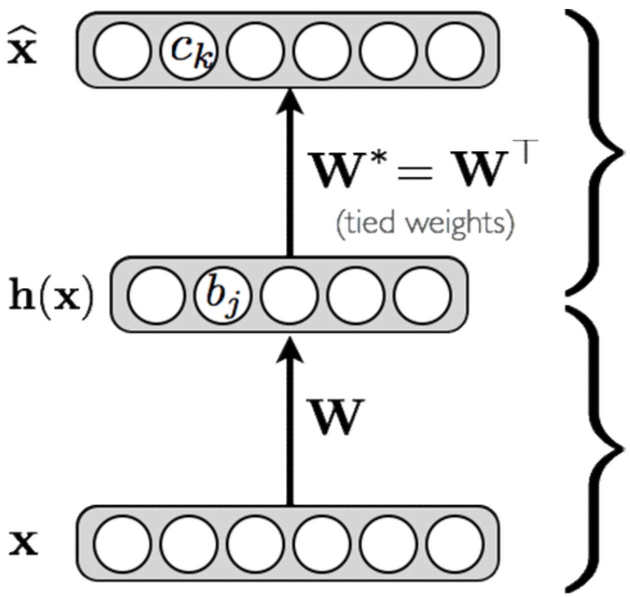
- Let us consider the following theorem:

- let $\mathbf{A}$ be the empirical covariance of data points
- Let $\mathbf{A} = \mathbf{U}\,\Sigma\,\mathbf{U}^{\top}$
- singular value decomposition
  - Σ is a diagonal matrix
  - $\mathbf{U}$ are orthonormal matrices (columns/rows are orthonormal vectors)

Linear Autoencoder (PCA)

- The optimal pair of encoder and decoder is given by:

$$\mathbf{h}(\mathbf{x}) = \underbrace{(\mathbf{U}_{\cdot,\leq k})^\top}_{\mathbf{W}} \mathbf{x} \qquad \widehat{\mathbf{x}} = \underbrace{\mathbf{U}_{\cdot,\leq k}}_{\mathbf{W}^*} \mathbf{h}(\mathbf{x})$$

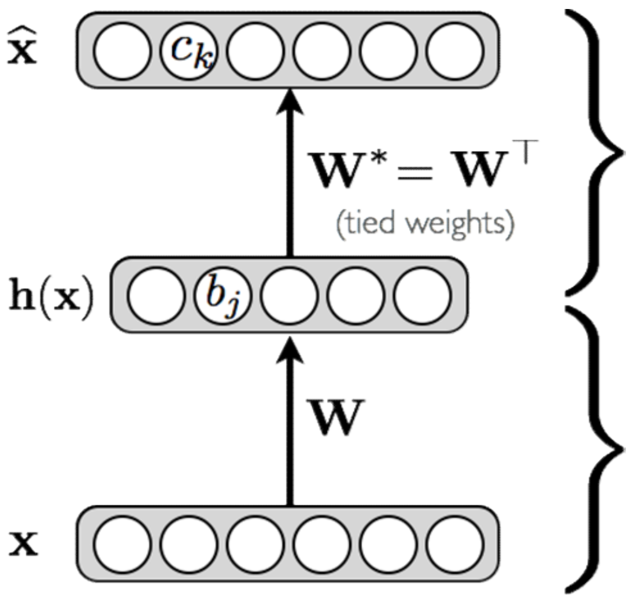


- for the sum of squared difference error
- for an auto-encoder with a linear decoder
- where optimality means “has the lowest training reconstruction error”

Optimality of the Linear Autoencoder

- So an optimal pair of encoder and decoder is

$$\mathbf{h}(\mathbf{x}) = \underbrace{(\mathbf{U}_{\cdot,\leq k})^\top}_{\mathbf{W}} \mathbf{x} \qquad\qquad \widehat{\mathbf{x}} = \underbrace{\mathbf{U}_{\cdot,\leq k}}_{\mathbf{W}^*} \mathbf{h}(\mathbf{x})$$



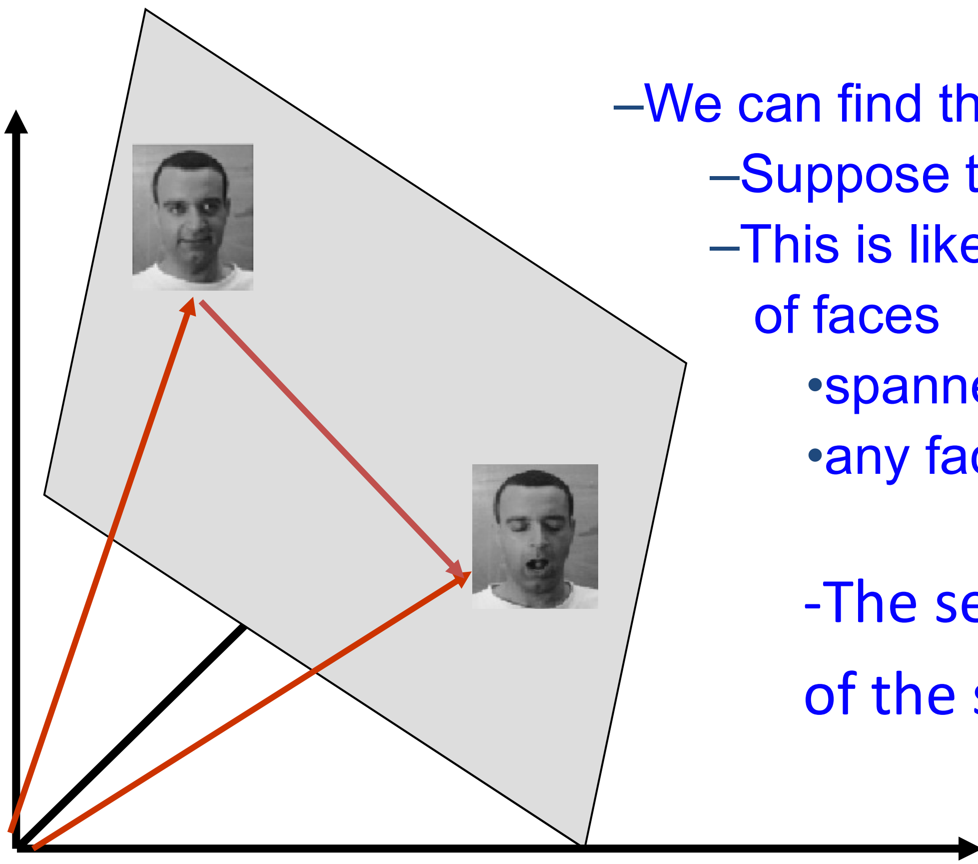
- If inputs are normalized as follows:

$$\mathbf{x}^{(t)} \leftarrow \frac{1}{\sqrt{T}}\left(\mathbf{x}^{(t)} - \frac{1}{T}\sum_{t'=1}^{T} \mathbf{x}^{(t')}\right)$$

  - encoder corresponds to Principal Component Analysis (PCA)
  - singular values and (left) vectors = the eigenvalues/vectors of covariance matrix

Example: Applications of PCA

- **Face Recognition**
- An image is a point in a high-dimensional space
  - An $N\ x\ M$ image is a point in R^{NM}
  - We can define vectors in this space



–We can find the best subspace using PCA

- –Suppose this is a K-dimensional subspace
- –This is like fitting a “hyper-plane” to the set of faces
  - •spanned by vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_K}$
  - •any face $\mathbf{x} \sim a_1\mathbf{v_1} + a_2\mathbf{v_2} + , \ldots, + a_K\mathbf{v_K}$

-The set of faces is a “subspace” of the set of images.

Application of PCA

- Let $F_1, F_2, \ldots, F_M$ be a set of training face images.
- Let F be their mean and $f_i = F_i$ – F
- Use principal components to compute the eigenvectors and eigenvalues of the covariance matrix of the $f_i's$
- Choose the vector u of most significant M eigenvectors to use as the basis.
- Each face (with F removed) is represented as a linear combination of eigenfaces $u = (u_1, u_2, u_3, u_4, u_5)$;
- $f_{27} = a_1 * u_1 + a_2 * u_2 + \ldots + a_5 * u_5$

unknown face image → convert to its eigenface representation → $A = (a_1, a_2, \ldots, a_m)$

Find the face class k that minimizes

$$||A - A_k||$$

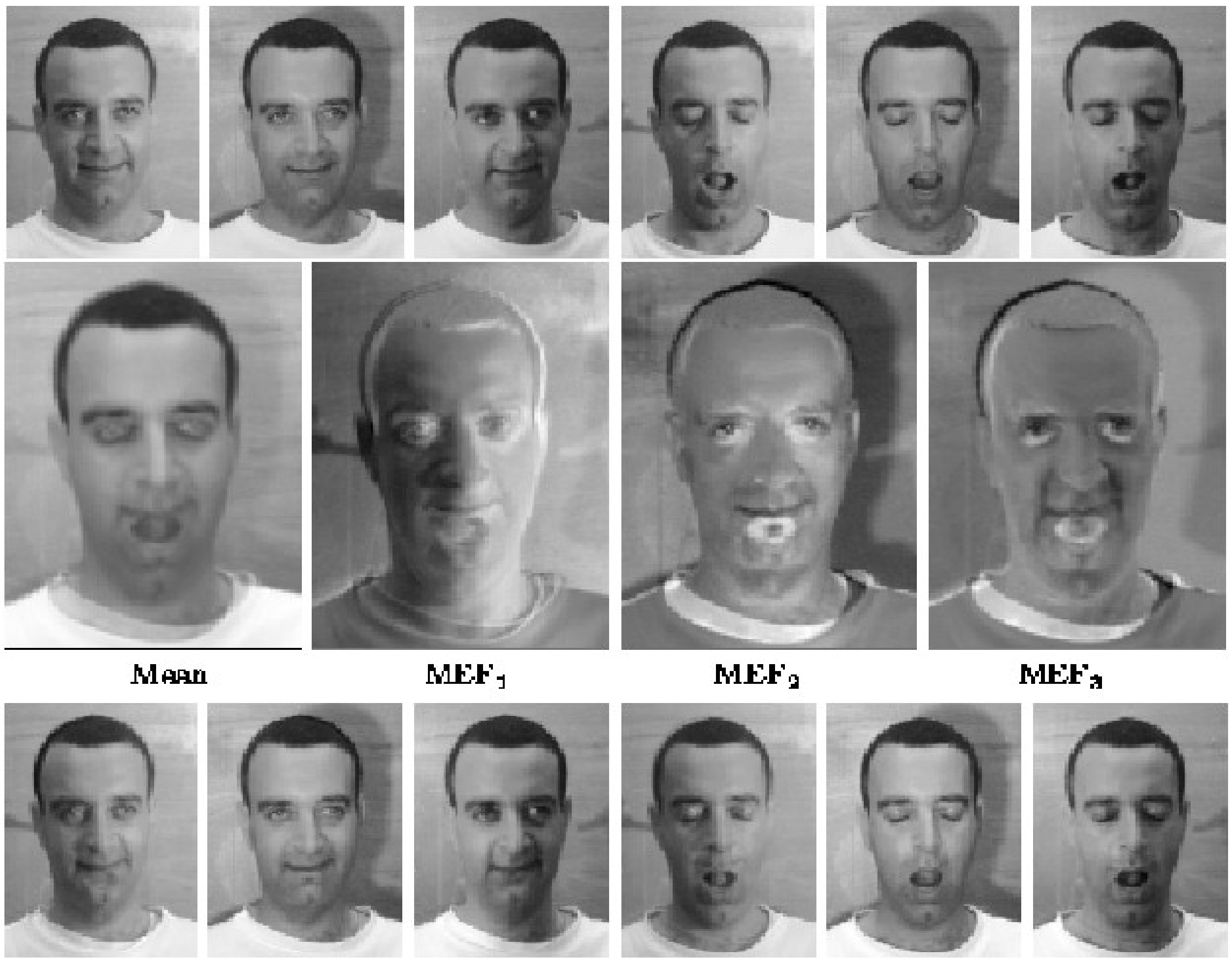
Application of PCA

training images

mean image

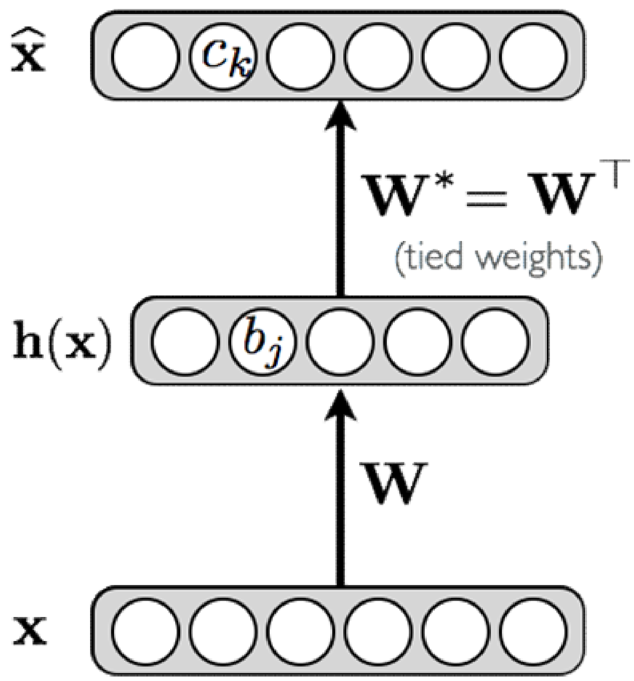
3 eigen-images

linear approxi-mations



Nonlinear Autoencoder (Example)

- Feed-forward neural network trained to reproduce its input at the output layer
- Nonlinear generalization of PCA



Decoder

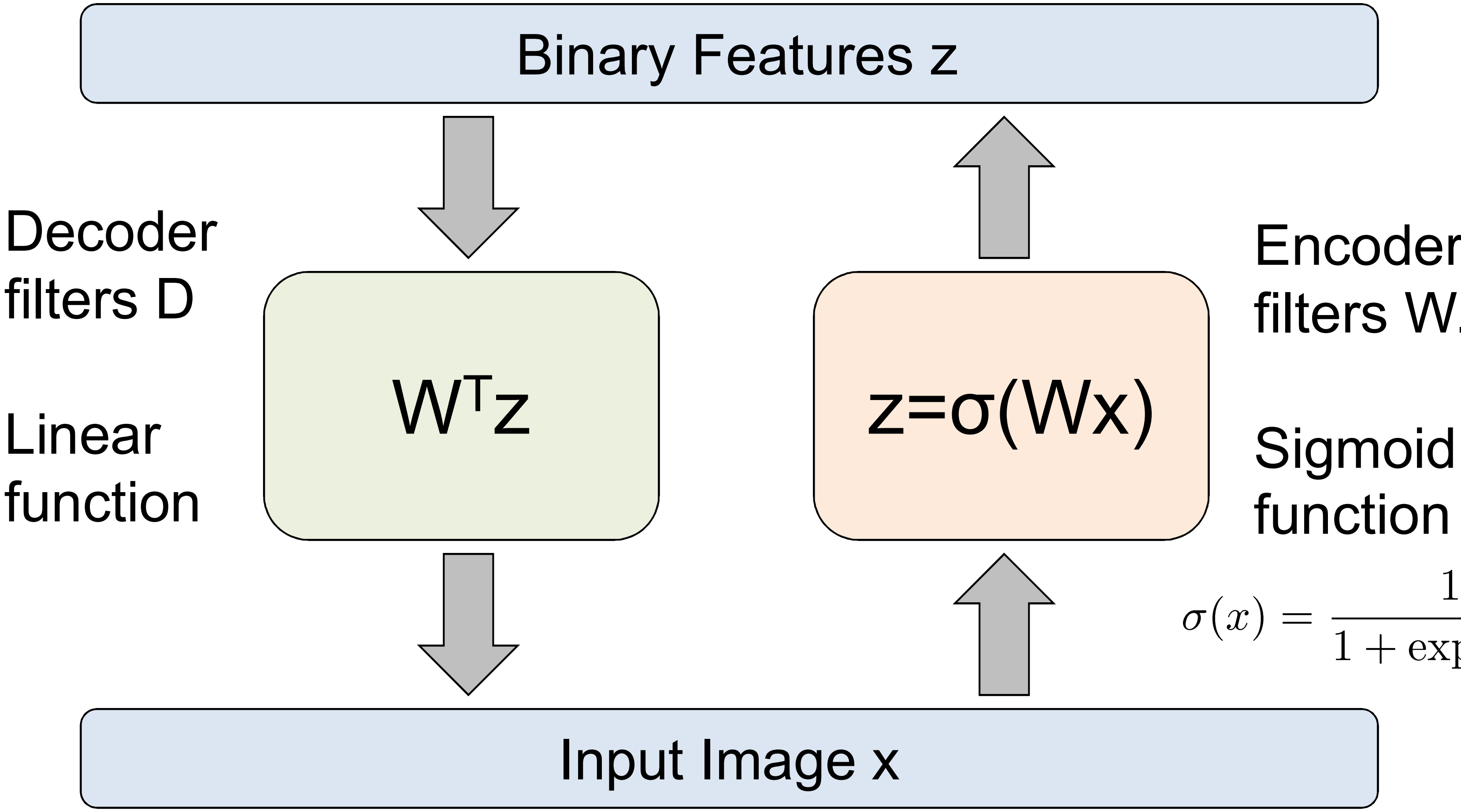
$$\begin{aligned}\hat{\mathbf{x}} &= \hat{\mathbf{a}}(\mathbf{x}) \\ &= \underbrace{\mathrm{sigm}}_{\text{For binary units}}(\mathbf{c} + \mathbf{W}^* \mathbf{h}(\mathbf{x}))\end{aligned}$$

Encoder

$$\begin{aligned}\mathbf{h}(\mathbf{x}) &= g(\mathbf{a}(\mathbf{x})) \\ &= \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})\end{aligned}$$

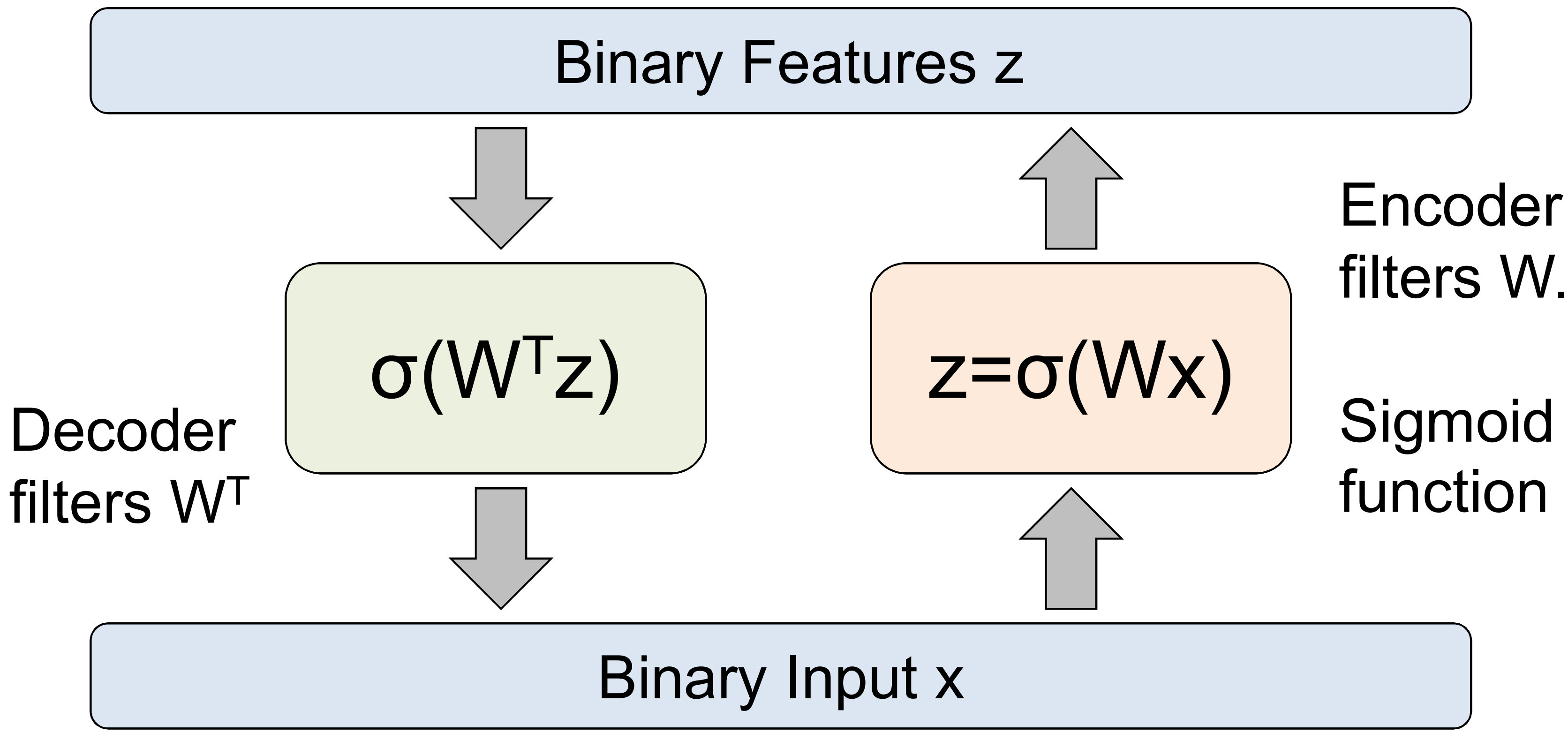
- In general it is hard to prove optimality results in nonlinear case.

Autoencoders (Example)



$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

Autoencoder Example



- Need additional constraints to avoid learning an identity.
- In general, Encoder and Decoder filters can be different.

Loss Function (Examples)

- Reproduce (reconstruct) the input at the output layer
- $f(x)$ is the output of autoencoder (model prediction)

- Loss function for binary inputs

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

  - Cross-entropy error function (reconstruction loss) $f(\mathbf{x}) \equiv \widehat{\mathbf{x}}$

- Loss function for real-valued inputs

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

  - sum of squared differences (reconstruction loss)
  - we use a linear activation function at the output

Loss Function

- Parameter gradients are obtained by back-propagating the gradient $\nabla_{\widehat{\mathbf{a}}(\mathbf{x}^{(t)})} l(f(\mathbf{x}^{(t)}))$ like in a regular network

  - Important: when using tied weights ($\mathbf{W}^* = \mathbf{W}^\top$), $\nabla_{\mathbf{W}} l(f(\mathbf{x}^{(t)}))$ is the sum of two gradients (Show this as an exercise).
  - this is because $\mathbf{W}$ is present in the encoder and in the decoder

Autoencoder example

Encoder architecture:

- Convolutional layer
  - 16 output channels , kernel size = 3, stride=3, padding=1
- ReLU
- MaxPooling
  - size = 2X2, stride=2
- Convolutional layer
  - 8 output channels, kernel size =3, stride=2, padding=1
- ReLU
- MaxPooling
  - size = 2, stride=1

Decoder architecture:

- Convolutional Transpose layer
  - 8 input channels , 16 output channels, kernel size=3, stride=2, padding=1
- ReLU
- Convolutional Transpose layer
  - 16 input chennels, 8 output channels, kernel size =5, stride=3, padding=1
- ReLU
- Convolutional Transpose layer
  - 8 input channels , 1 output channel, kernel size=2, stride=2, padding=1
- Tanh

Example: MNIST

100 samples from test data set

100 generated samples using Autoencoder

- **20 epochs, batch size = 128, Adam optimizer, learning rate = 0.001**
- **Normalized images with mean 0.5 and std equals to 0.5**

Autoencoder

- How to extract meaningful hidden features?

1. **Undercomplete Representation**
2. **Injecting noise to the input (Denoising Autoencoder)**
3. **Penalizing the solution (Contractive Autoencoders)**

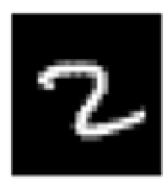
Undercomplete Representation

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

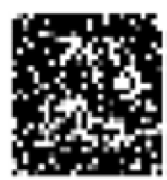
- hidden layer “compresses” the input
- may compress well only for the training distribution

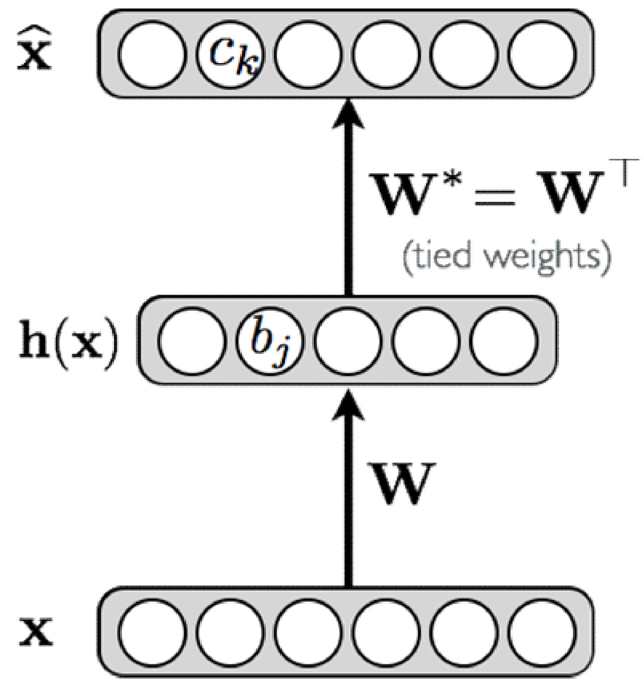
• Hidden units will be

- good features for the training distribution



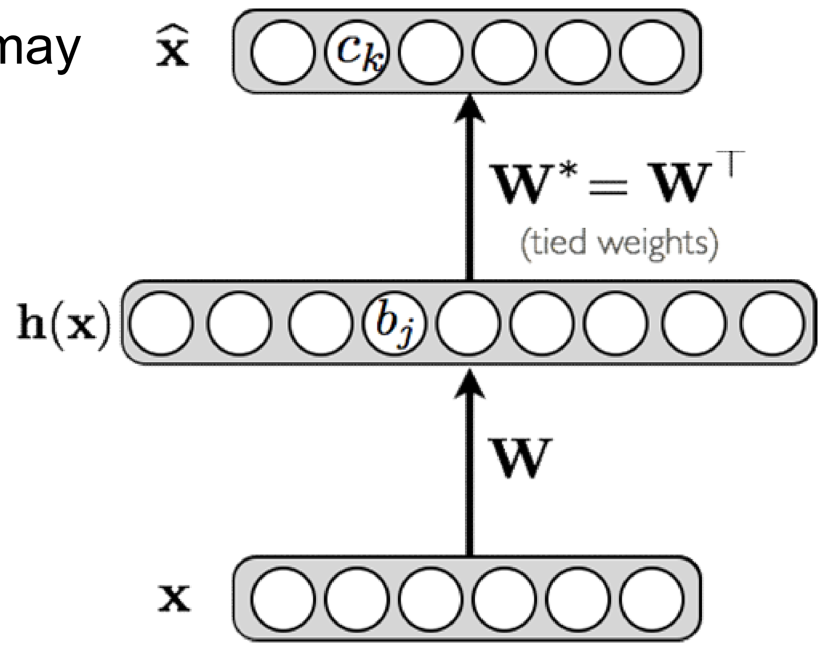
- may not be robust to other types of input





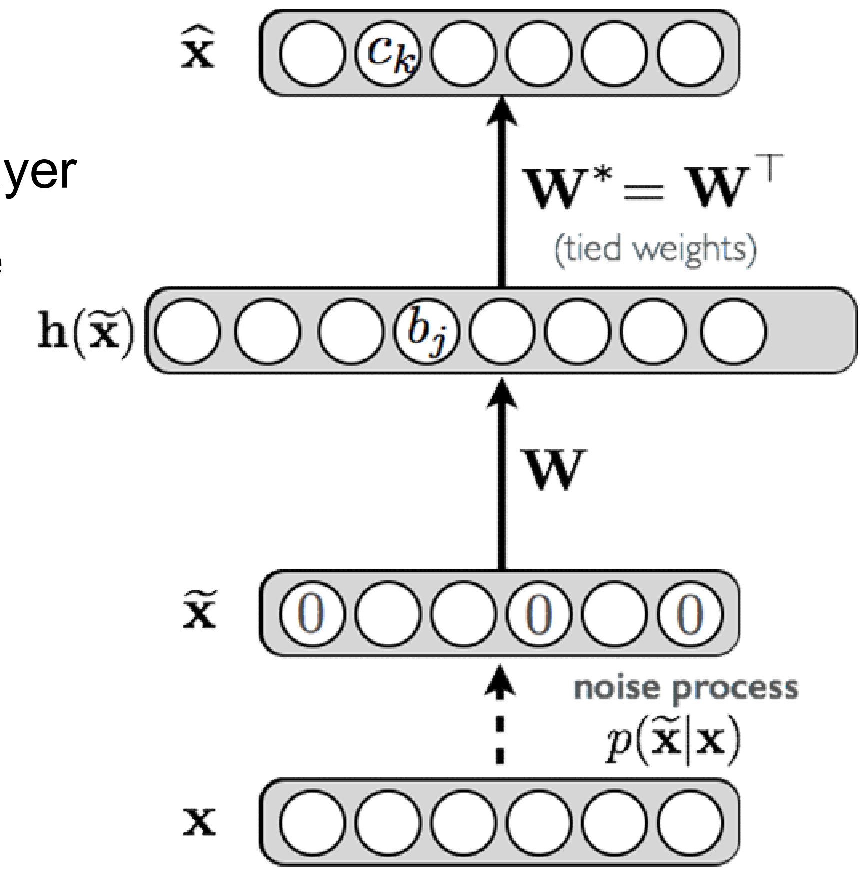
Overcomplete Representation

- Hidden layer is overcomplete if greater than the input layer
  - no compression in hidden layer
  - In some cases each hidden unit may copy a different input component
- No guarantee that the hidden units will extract meaningful structure



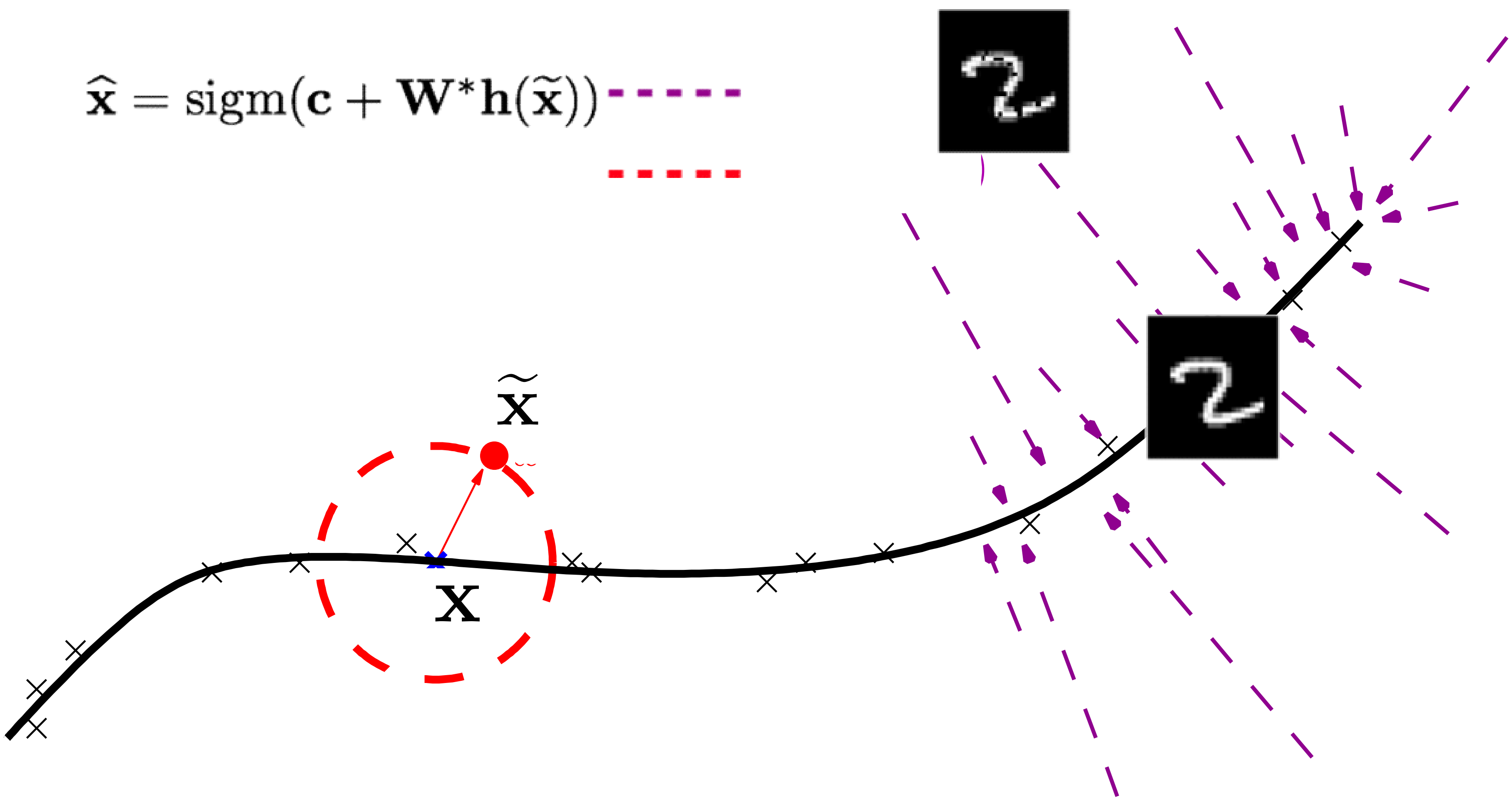
Denoising Autoencoder

- Idea: representation should be robust to introduction of noise:
  - random assignment of subset of inputs to 0, with probability ν
  - Similar to dropouts on the input layer
  - Similar idea for Gaussian additive noise
- Reconstruction $\widehat{\mathbf{x}}$ computed from the corrupted input $\widetilde{\mathbf{x}}$
- Loss function compares $\widehat{\mathbf{x}}$ reconstruction with the noiseless input $\mathbf{x}$



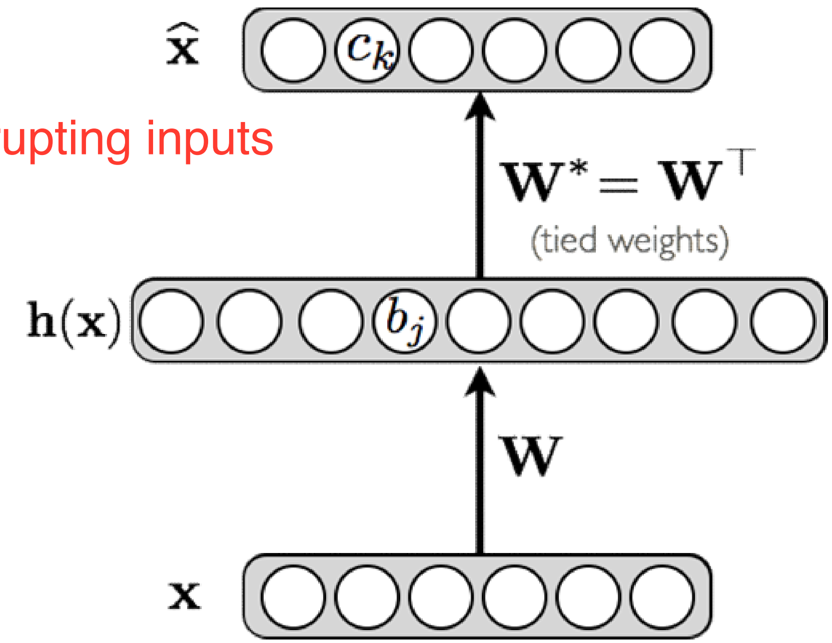
(Vincent et.al., ICML 2008)

Denoising Autoencoder



Contractive Autoencoder

- Alternative approach to avoid uninteresting solutions
  - ➢ add an explicit term in the loss that penalizes that solution



You want denoising behavior without explicitly corrupting inputs

- We wish to extract features that only reflect variations observed in the training set
  - ➢ we'd like to be invariant to the other variations

Contractive Autoencoder

- Consider the following loss function:

$$\underbrace{l(f(\mathbf{x}^{(t)}))}_{\text{Reconstruction Loss}} + \lambda \underbrace{||\nabla_{\mathbf{x}^{(t)}} \mathbf{h}(\mathbf{x}^{(t)})||_F^2}_{\text{Jacobian of Encoder}}$$

- Example for binary observations:

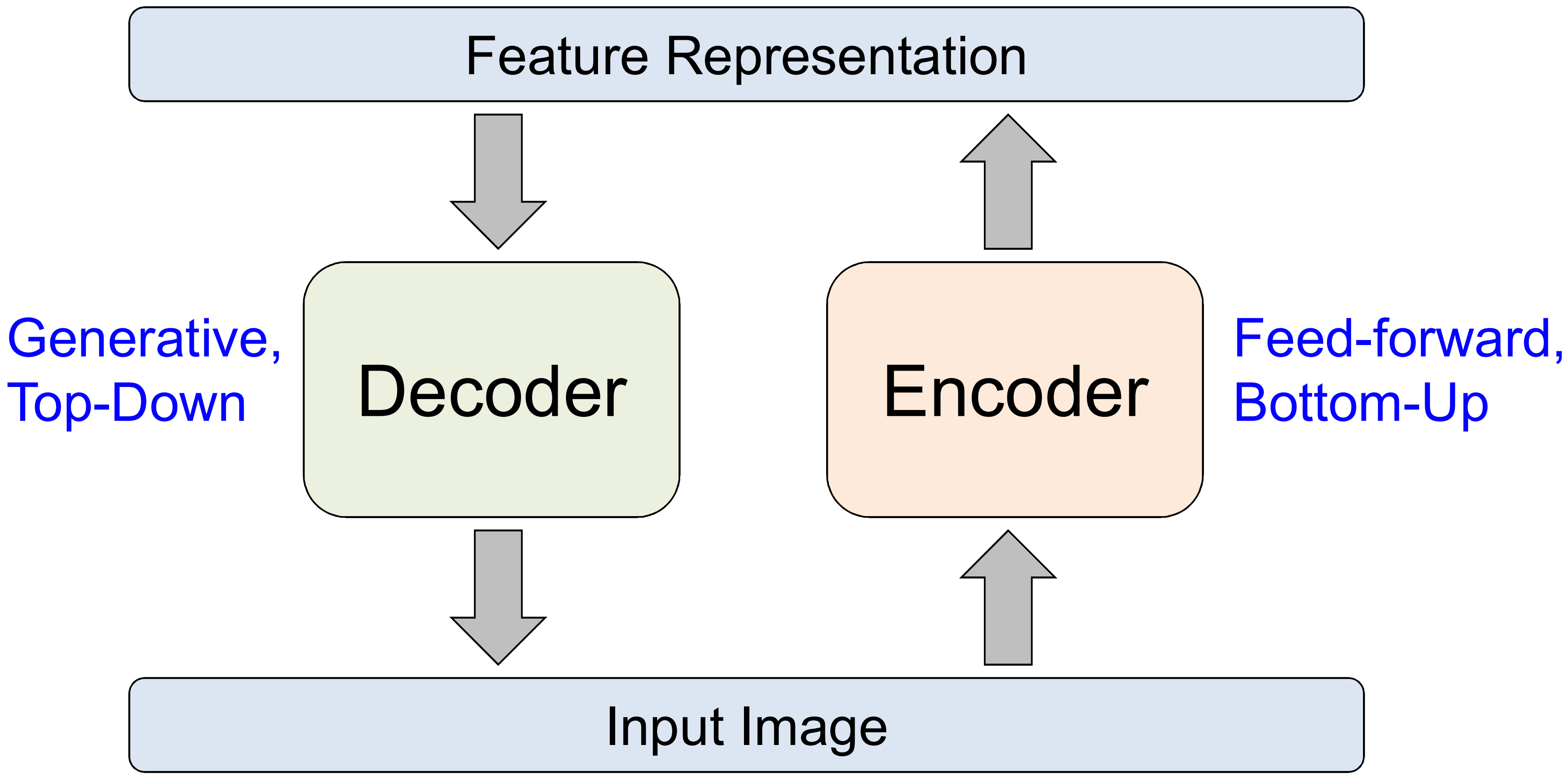
$$l(f(\mathbf{x}^{(t)})) = -\sum_k \left(x_k^{(t)} \log(\widehat{x}_k^{(t)}) + (1 - x_k^{(t)}) \log(1 - \widehat{x}_k^{(t)}) \right)$$

$$||\nabla_{\mathbf{x}^{(t)}} \mathbf{h}(\mathbf{x}^{(t)})||_F^2 = \sum_j \sum_k \left(\frac{\partial h(\mathbf{x}^{(t)})_j}{\partial x_k^{(t)}} \right)^2$$

Autoencoder attempts to preserve all information

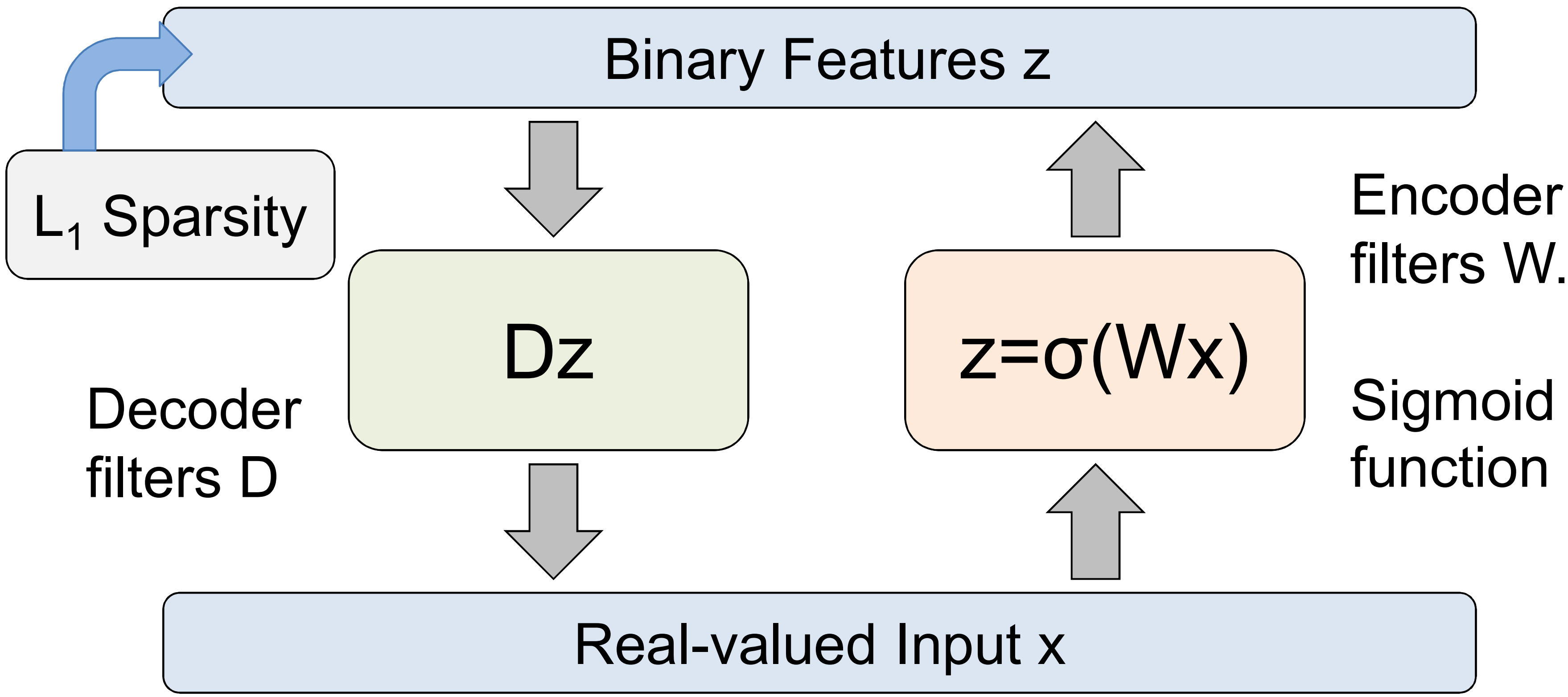
Encoder throws away all information

Autoencoder



- Details of what goes insider the encoder and decoder matter!

Predictive Sparse Decomposition

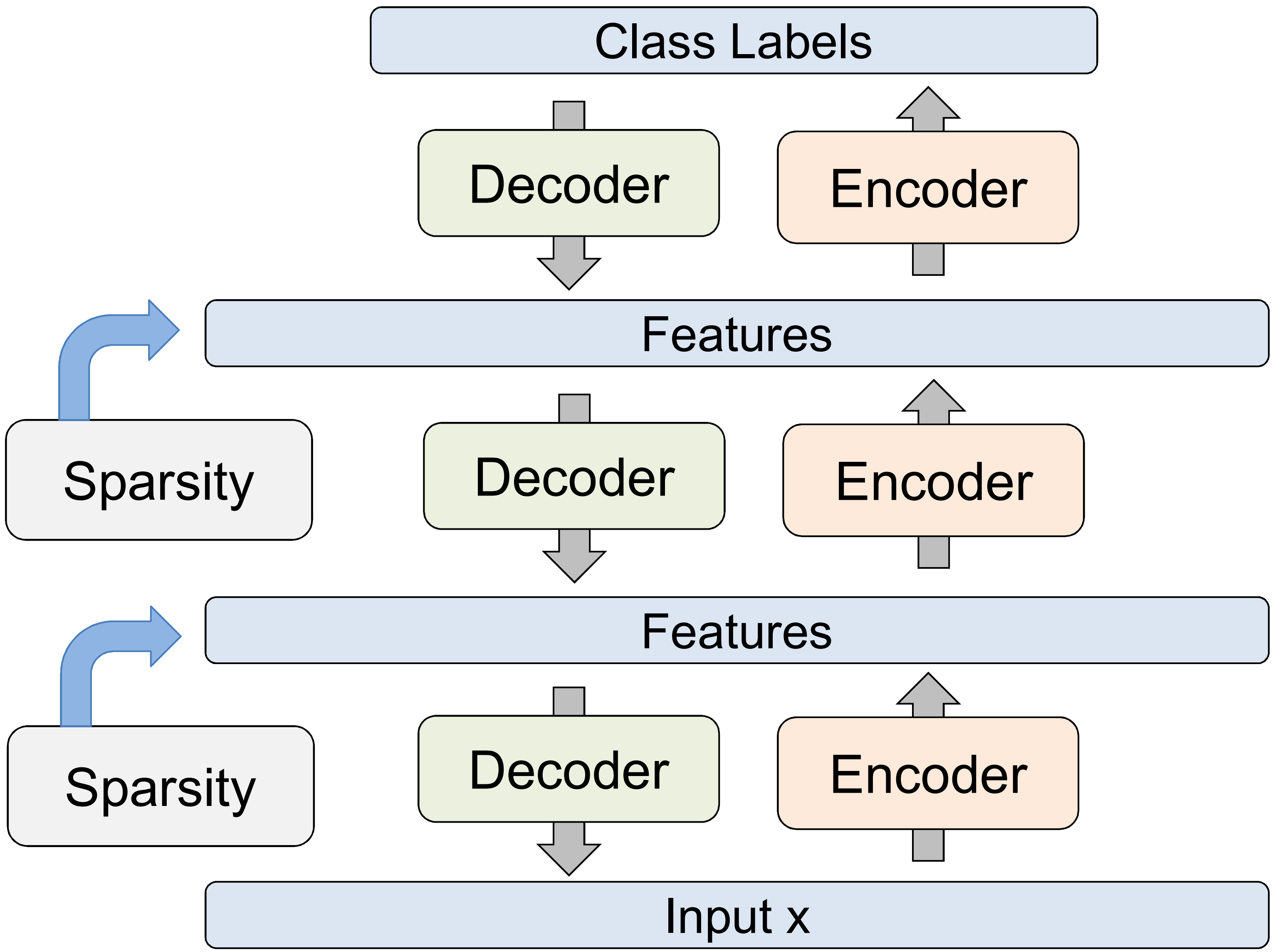


At training time

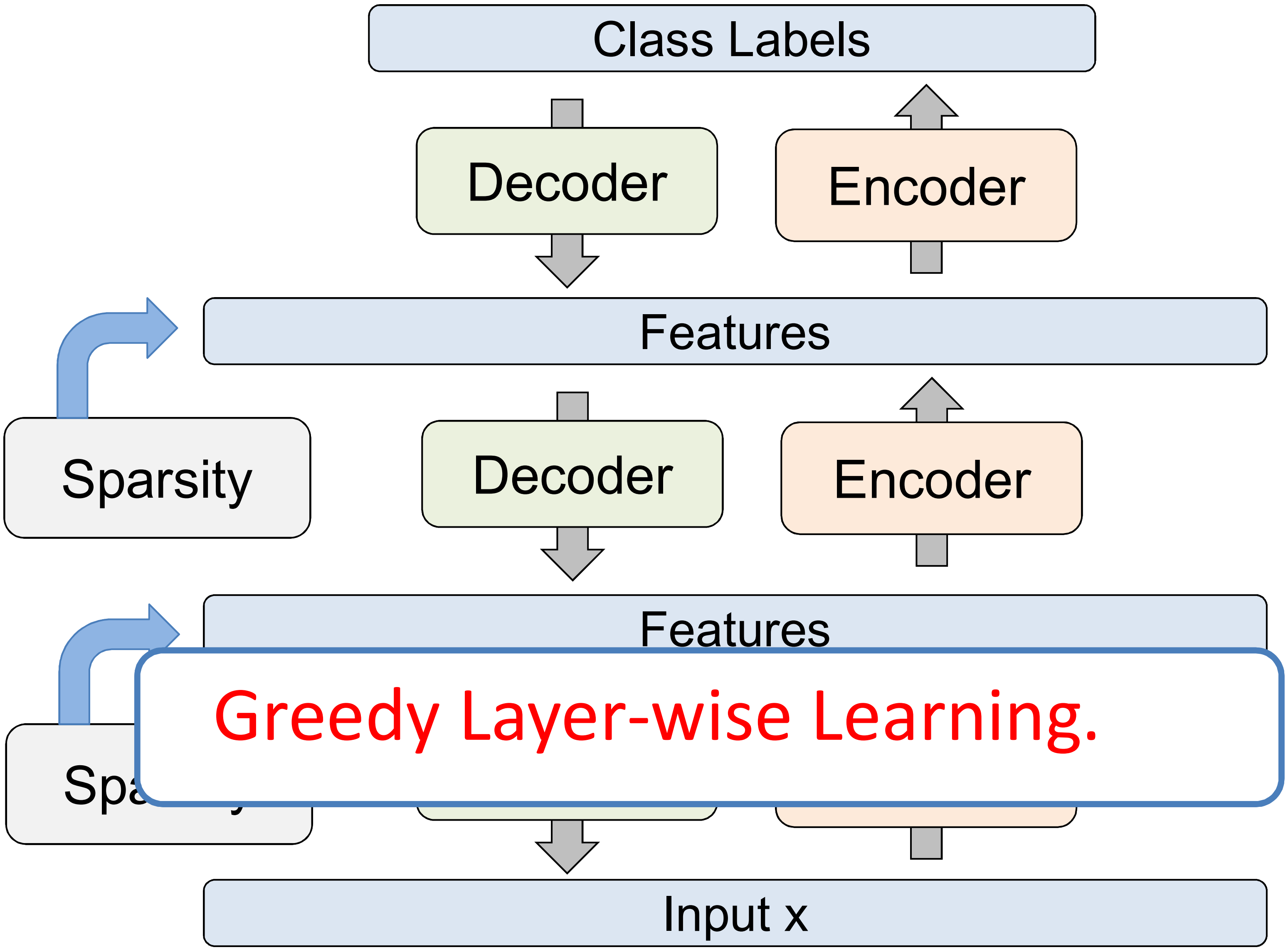
$$\min_{D,W,\mathbf{z}} \underbrace{||D\mathbf{z} - \mathbf{x}||_2^2 + \lambda|\mathbf{z}|_1}_{\text{Decoder}} + \underbrace{||\sigma(W\mathbf{x}) - \mathbf{z}||_2^2}_{\text{Encoder}}$$

Kavukcuoglu et al., '09

Stacked Autoencoders



Stacked Autoencoders



Greedy Layer-wise Learning.