Neural networks

Autoencoder - definition

UNSUPERVISED LEARNING

Topics: unsupervised learning

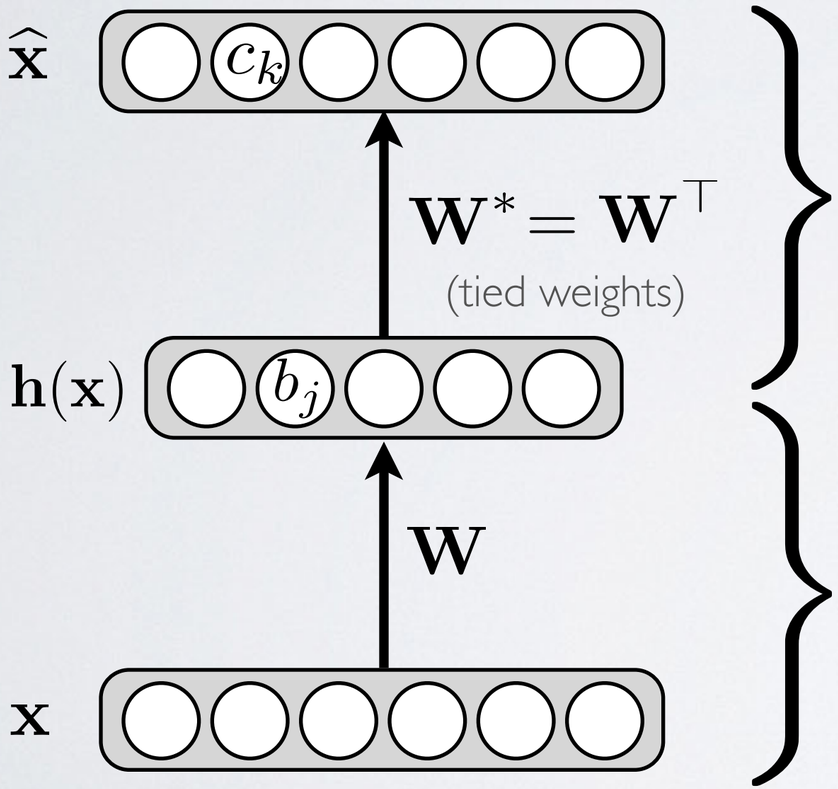
- Unsupervised learning: only use the inputs $\mathbf{x}^{(t)}$ for learning
  - automatically extract meaningful features for your data
  - leverage the availability of unlabeled data
  - add a data-dependent regularizer to training

- We will see a number of neural networks for unsupervised learning
  - **autoencoders**
  - autoregressive models
  - variational autoencoders
  - generative adversarial networks

AUTOENCODER

Topics: autoencoder, encoder, decoder, tied weights

- Feed-forward neural network trained to reproduce its input at the output layer



Decoder

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

Encoder

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

AUTOENCODER

Topics: loss function

- For binary inputs:

$$f(\mathbf{x}) \equiv \hat{\mathbf{x}}$$

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

  - cross-entropy (more precisely: sum of Bernoulli cross-entropies)

- For real-valued inputs:

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

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

AUTOENCODER

Topics: loss function gradient

- For both cases, the gradient $\nabla_{\hat{\mathbf{a}}(\mathbf{x}^{(t)})} l(f(\mathbf{x}^{(t)}))$ has a very simple form:

$$f(\mathbf{x}) \equiv \hat{\mathbf{x}}$$

$$\nabla_{\hat{\mathbf{a}}(\mathbf{x}^{(t)})} l(f(\mathbf{x}^{(t)})) = \hat{\mathbf{x}}^{(t)} - \mathbf{x}^{(t)}$$

- Parameter gradients are obtained by backpropagating the gradient $\nabla_{\hat{\mathbf{a}}(\mathbf{x}^{(t)})} l(f(\mathbf{x}^{(t)}))$ 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 !
    - this is because $\mathbf{W}$ is present in the encoder **and** in the decoder

AUTOENCODER

Topics: adaptation to the type of input

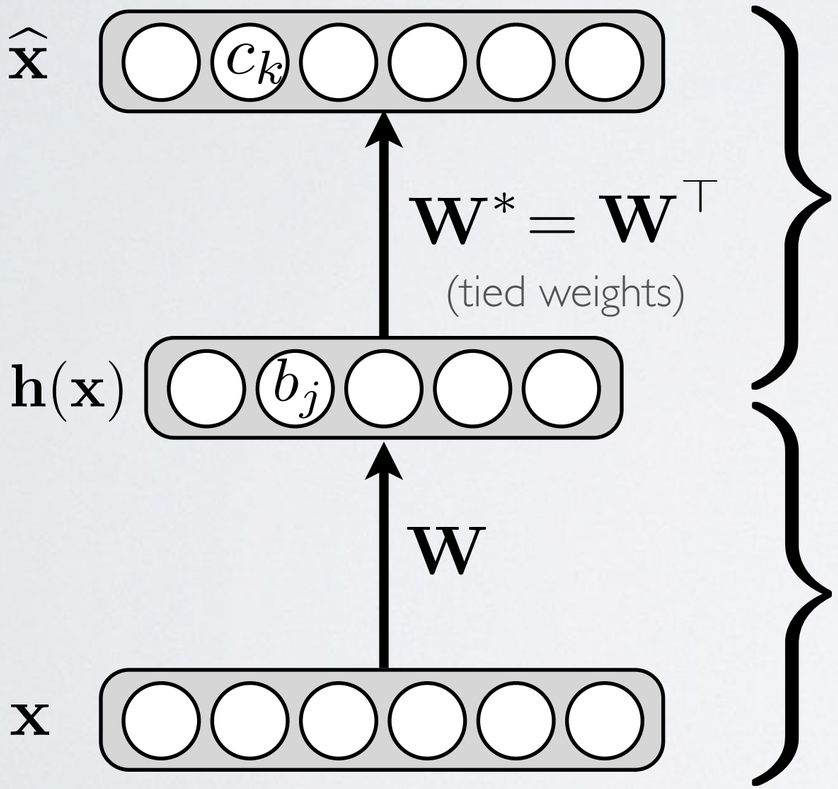
$$f(\mathbf{x}) \equiv \hat{\mathbf{x}}$$

- Recipe to adapt an autoencoder to a new type of input
  - choose a joint distribution $p(\mathbf{x}|\boldsymbol{\mu})$ over the inputs
    - $\boldsymbol{\mu}$ is the vector of parameters of that distribution
  - choose the relationship between $\boldsymbol{\mu}$ and the hidden layer $\mathbf{h}(\mathbf{x})$
  - use $l(f(\mathbf{x})) = -\log p(\mathbf{x}|\boldsymbol{\mu})$ as the loss function
- Example: we get the sum of squared distance by
  - choosing a Gaussian distribution with mean $\boldsymbol{\mu}$ and identity covariance for $p(\mathbf{x}|\boldsymbol{\mu}) = \frac{1}{(2\pi)^{D/2}} \exp(-\frac{1}{2}\sum_k (x_k - \mu_k)^2)$
  - choosing $\boldsymbol{\mu} = \mathbf{c} + \mathbf{W}^* \mathbf{h}(\mathbf{x})$

AUTOENCODER

Topics: autoencoder, encoder, decoder, tied weights

- Feed-forward neural network trained to reproduce its input at the output layer



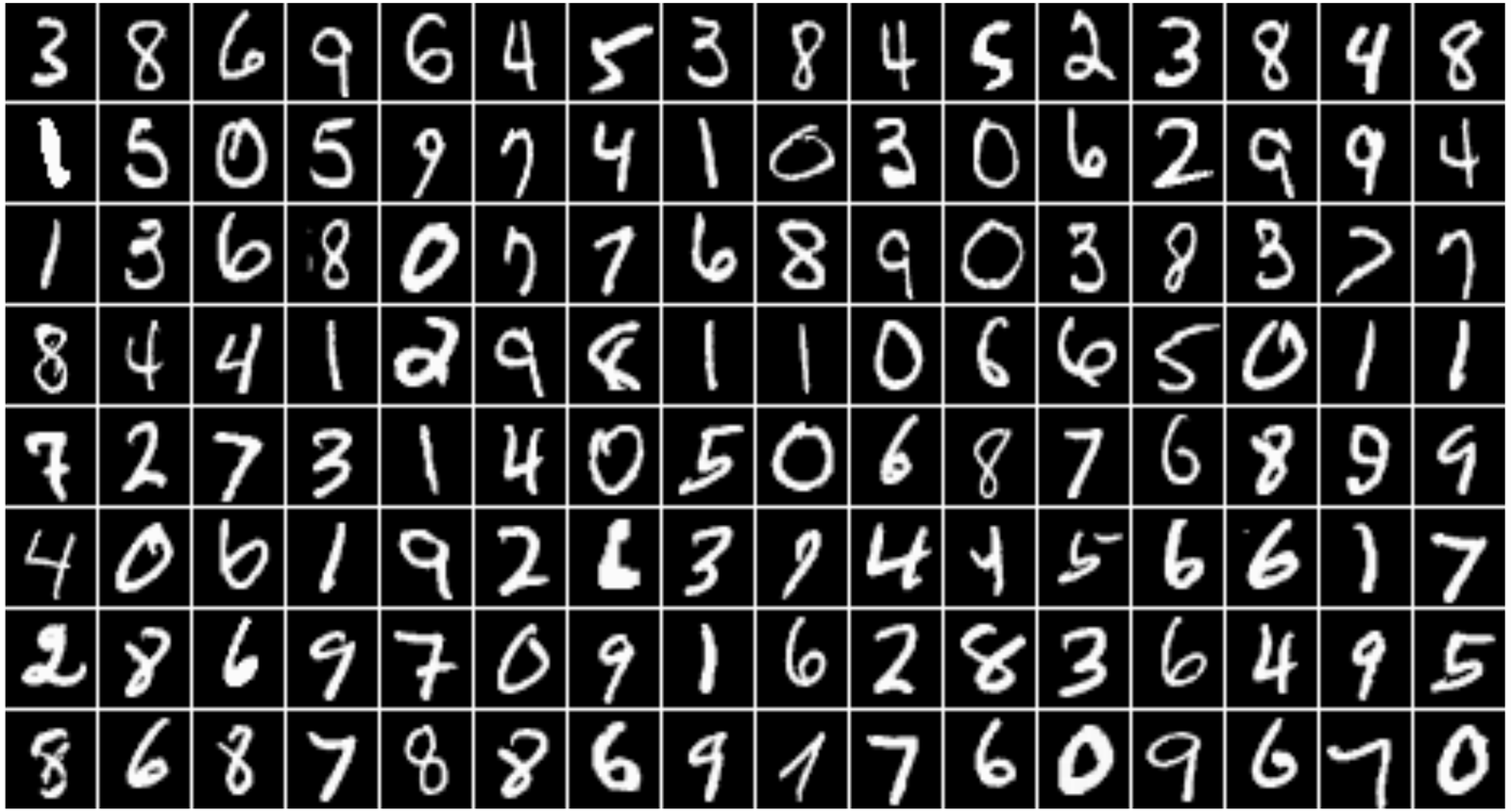
Decoder

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

Encoder

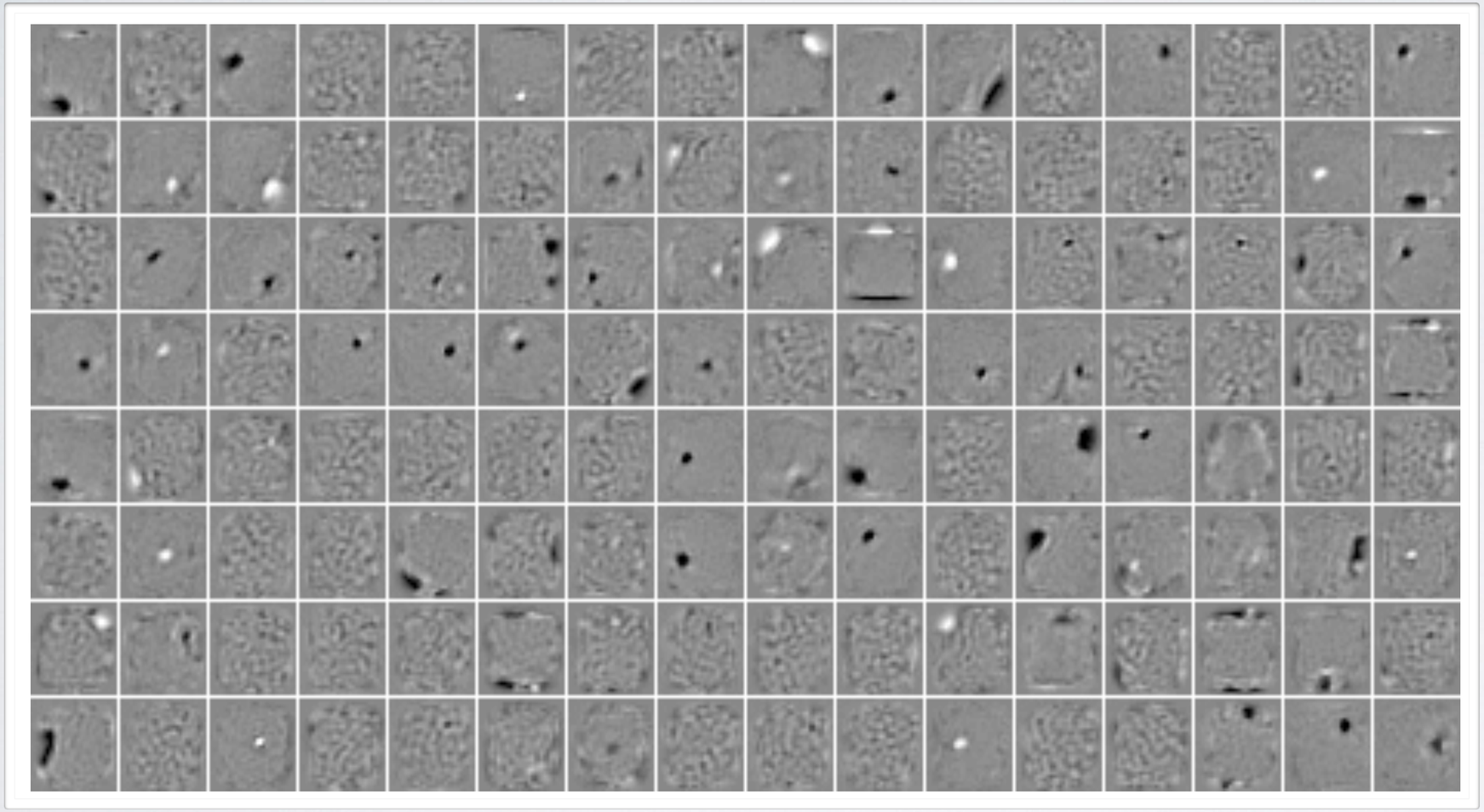
$$\begin{aligned}\mathbf{h}(\mathbf{x}) &= g(\mathbf{a}(\mathbf{x})) \\ &= \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})\end{aligned}$$

EXAMPLE OF DATA SET: MNIST



FILTERS (AUTOENCODER)

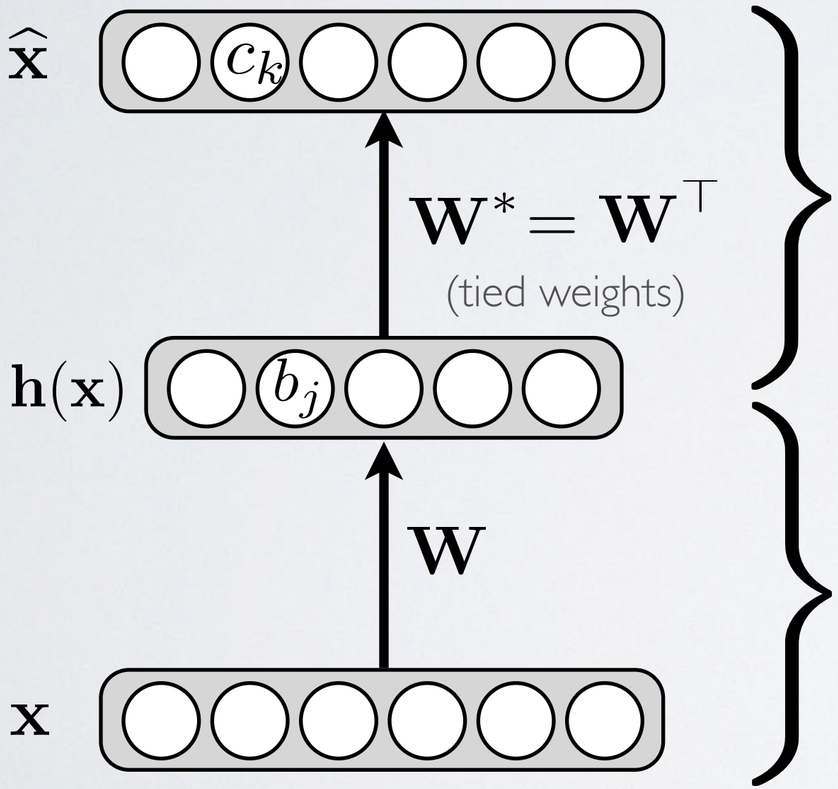
(Larochelle et al., JMLR2009)



AUTOENCODER

Topics: autoencoder, encoder, decoder, tied weights

- Feed-forward neural network trained to reproduce its input at the output layer



Decoder

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

Encoder

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

AUTOENCODER

Topics: optimality of a linear autoencoder

- To do the proof, we need the following theorem:
  - let $\mathbf{A}$ be any matrix, with singular value decomposition $\mathbf{A} = \mathbf{U}\,\Sigma\,\mathbf{V}^\top$
    - Σ is a diagonal matrix
    - $\mathbf{U}, \mathbf{V}$ are orthonormal matrices (columns/rows are orthonormal vectors)
  - let $\mathbf{U}_{\cdot,\leq k}\,\Sigma_{\leq k,\leq k}\,\mathbf{V}^\top_{\cdot,\leq k}$ be the decomposition where we keep only the ***k*** largest singular values
  - then, the matrix $\mathbf{B}$ of rank ***k*** that is closest to $\mathbf{A}$:

$$\mathbf{B}^* = \underset{\mathbf{B}\ \text{s.t.}\ \text{rank}(\mathbf{B})=k}{\arg\min} ||\mathbf{A} - \mathbf{B}||_F$$

  is $\mathbf{B}^* = \mathbf{U}_{\cdot,\leq k}\,\Sigma_{\leq k,\leq k}\,\mathbf{V}^\top_{\cdot,\leq k}$

$$\min_{\theta} \sum_{t} \frac{1}{2} \sum_{i} (x_i^{(t)} - \widehat{x}_i^{(t)})^2$$

Sketch of proof

matrix where columns are $\mathbf{x}^{(t)}$

$$\min_{\theta} \sum_t \frac{1}{2} \sum_i (x_i^{(t)} - \underbrace{\widehat{x}_i^{(t)}}_{\text{based on linear decoder}})^2 \quad \geq \quad \min_{\mathbf{W}^*, \mathbf{h}(\mathbf{X})} \frac{1}{2} ||\overbrace{\mathbf{X}}^{\text{matrix where columns are } \mathbf{x}^{(t)}} - \mathbf{W}^* \underbrace{\mathbf{h}(\mathbf{X})}_{\text{matrix of all hidden layers (could be any encoder)}}||_F^2$$

Sketch of proof

$$\min_{\theta} \sum_t \frac{1}{2} \sum_i (x_i^{(t)} - \underbrace{\widehat{x}_i^{(t)}}_{\text{based on linear decoder}})^2 \quad \geq \quad \min_{\mathbf{W}^*, \mathbf{h}(\mathbf{X})} \frac{1}{2} ||\overbrace{\mathbf{X}}^{\text{matrix where columns are } \mathbf{x}^{(t)}} - \mathbf{W}^* \underbrace{\mathbf{h}(\mathbf{X})}_{\text{matrix of all hidden layers (could be any encoder)}}||_F^2$$

$$\arg\min_{\mathbf{W}^*, \mathbf{h}(\mathbf{X})} \frac{1}{2} ||\mathbf{X} - \mathbf{W}^* \mathbf{h}(\mathbf{X})||_F^2 \quad = \quad \left(\mathbf{W}^* \leftarrow \mathbf{U}_{\cdot, \leq k}\, \Sigma_{\leq k, \leq k}, \;\; \mathbf{h}(\mathbf{X}) \leftarrow \mathbf{V}_{\cdot, \leq k}^\top\right)$$

based on previous theorem, where $\mathbf{X} = \mathbf{U}\, \Sigma\, \mathbf{V}^\top$ and ***k*** is the hidden layer size

Let's show $\mathbf{h}(\mathbf{X})$ is a linear encoder:

$$\mathbf{h}(\mathbf{X}) \quad = \quad \mathbf{V}_{\cdot, \leq k}^\top$$

Sketch of proof

matrix where columns are $\mathbf{x}^{(t)}$

$$\min_{\theta} \sum_t \frac{1}{2} \sum_i (x_i^{(t)} - \underbrace{\widehat{x}_i^{(t)}}_{\text{based on linear decoder}})^2 \quad \geq \quad \min_{\mathbf{W}^*, \mathbf{h}(\mathbf{X})} \frac{1}{2} ||\overbrace{\mathbf{X}}^{\text{matrix of all hidden layers (could be any encoder)}} - \mathbf{W}^* \underbrace{\mathbf{h}(\mathbf{X})}|_F^2$$

$$\arg\min_{\mathbf{W}^*, \mathbf{h}(\mathbf{X})} \frac{1}{2} ||\mathbf{X} - \mathbf{W}^* \mathbf{h}(\mathbf{X})||_F^2 \quad = \quad \left(\mathbf{W}^* \leftarrow \mathbf{U}_{\cdot, \leq k}\, \Sigma_{\leq k, \leq k},\ \ \mathbf{h}(\mathbf{X}) \leftarrow \mathbf{V}_{\cdot, \leq k}^\top\right)$$

based on previous theorem, where $\mathbf{X} = \mathbf{U}\, \Sigma\, \mathbf{V}^\top$ and ***k*** is the hidden layer size

Let's show $\mathbf{h}(\mathbf{X})$ is a linear encoder:

$$\begin{aligned}
\mathbf{h}(\mathbf{X}) &= \mathbf{V}_{\cdot, \leq k}^\top \\
&= \mathbf{V}_{\cdot, \leq k}^\top\, (\mathbf{X}^\top\, \mathbf{X})^{-1}\, (\mathbf{X}^\top\, \mathbf{X}) && \leftarrow \text{multiplying by identity} \\
&= \mathbf{V}_{\cdot, \leq k}^\top\, (\mathbf{V}\, \Sigma^\top\, \mathbf{U}^\top\, \mathbf{U}\, \Sigma\, \mathbf{V}^\top)^{-1}\, (\mathbf{V}\, \Sigma^\top\, \mathbf{U}^\top\, \mathbf{X}) && \leftarrow \text{replace with SVD} \\
&= \mathbf{V}_{\cdot, \leq k}^\top\, (\mathbf{V}\, \Sigma^\top\, \Sigma\, \mathbf{V}^\top)^{-1}\, \mathbf{V}\, \Sigma^\top\, \mathbf{U}^\top\, \mathbf{X} && \leftarrow \mathbf{U}^\top\, \mathbf{U} = \mathbf{I} \text{ (orthonormal)} \\
&= \mathbf{V}_{\cdot, \leq k}^\top\, \mathbf{V}\, (\Sigma^\top\, \Sigma)^{-1}\, \mathbf{V}^\top\, \mathbf{V}\, \Sigma^\top\, \mathbf{U}^\top\, \mathbf{X} && \leftarrow \mathbf{V}(\Sigma^\top\Sigma)^{-1}\mathbf{V}^\top\, \mathbf{V}\Sigma^\top\Sigma\mathbf{V}^\top = \mathbf{I} \\
&= \mathbf{V}_{\cdot, \leq k}^\top\, \mathbf{V}\, (\Sigma^\top\, \Sigma)^{-1}\, \Sigma^\top\, \mathbf{U}^\top\, \mathbf{X} && \leftarrow \mathbf{V}^\top\, \mathbf{V} = \mathbf{I} \text{ (orthonormal)} \\
&= \mathbf{I}_{\leq k, \cdot}\, (\Sigma^\top\, \Sigma)^{-1}\, \Sigma^\top\, \mathbf{U}^\top\, \mathbf{X} && \leftarrow \text{idem} \\
&= \mathbf{I}_{\leq k, \cdot}\, \Sigma^{-1}\, (\Sigma^\top)^{-1}\, \Sigma^\top\, \mathbf{U}^\top\, \mathbf{X} && \leftarrow (\Sigma^\top\Sigma)^{-1} = \Sigma^{-1}(\Sigma^\top)^{-1} \\
&= \mathbf{I}_{\leq k, \cdot}\, \Sigma^{-1}\, \mathbf{U}^\top\, \mathbf{X} \\
&= \underbrace{\Sigma_{\leq k, \leq k}^{-1}\, (\mathbf{U}_{\cdot, \leq k})^\top}_{\text{this is a linear encoder}}\, \mathbf{X} && \leftarrow \text{multiplying by } \mathbf{I}_{\leq k, \cdot} \text{ selects the } k \text{ first rows}
\end{aligned}$$

AUTOENCODER

Topics: optimality of a linear autoencoder

- So an optimal pair of encoder and decoder is

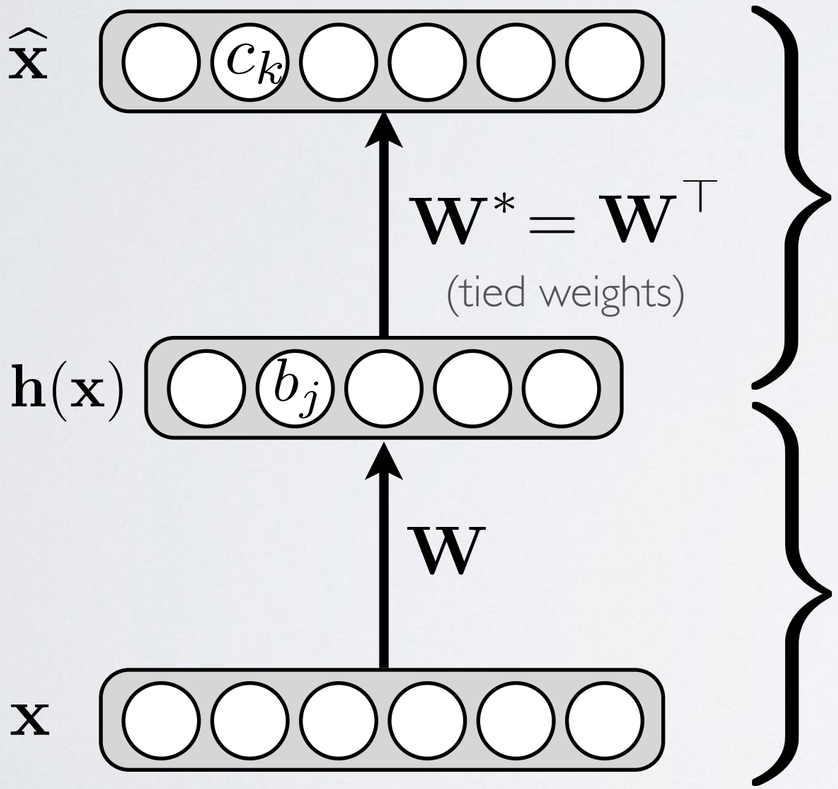
$$\mathbf{h}(\mathbf{x}) = \underbrace{\left(\Sigma^{-1}_{\leq k,\leq k} \left(\mathbf{U}_{\cdot,\leq k}\right)^\top\right)}_{\mathbf{W}} \mathbf{x} \qquad \hat{\mathbf{x}} = \underbrace{\left(\mathbf{U}_{\cdot,\leq k}\, \Sigma_{\leq k,\leq k}\right)}_{\mathbf{W}^*} \mathbf{h}(\mathbf{x})$$

  - for the sum of squared difference error
  - for an autoencoder with a linear decoder
  - where optimality means "has the lowest training reconstruction error"
- 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

AUTOENCODER

Topics: autoencoder, encoder, decoder, tied weights

- Feed-forward neural network trained to reproduce its input at the output layer



Decoder

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

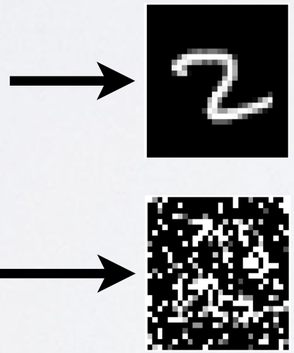
Encoder

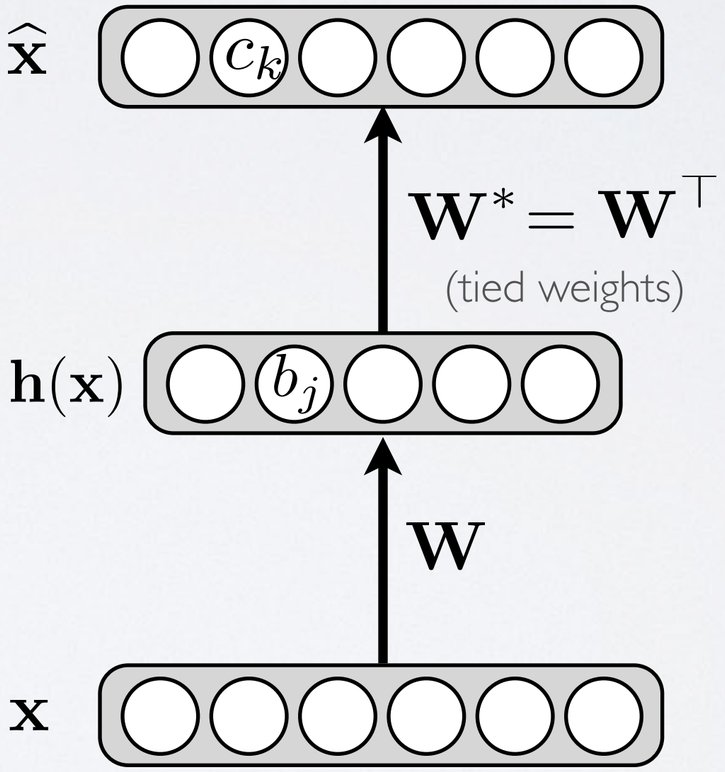
$$\begin{aligned}\mathbf{h}(\mathbf{x}) &= g(\mathbf{a}(\mathbf{x})) \\ &= \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})\end{aligned}$$

UNDERCOMPLETE HIDDEN LAYER

Topics: undercomplete representation

- Hidden layer is undercomplete if smaller than the input layer
  - hidden layer "compresses" the input
  - will compress well only for the training distribution
- Hidden units will be
  - good features for the training distribution
  - but bad for other types of input

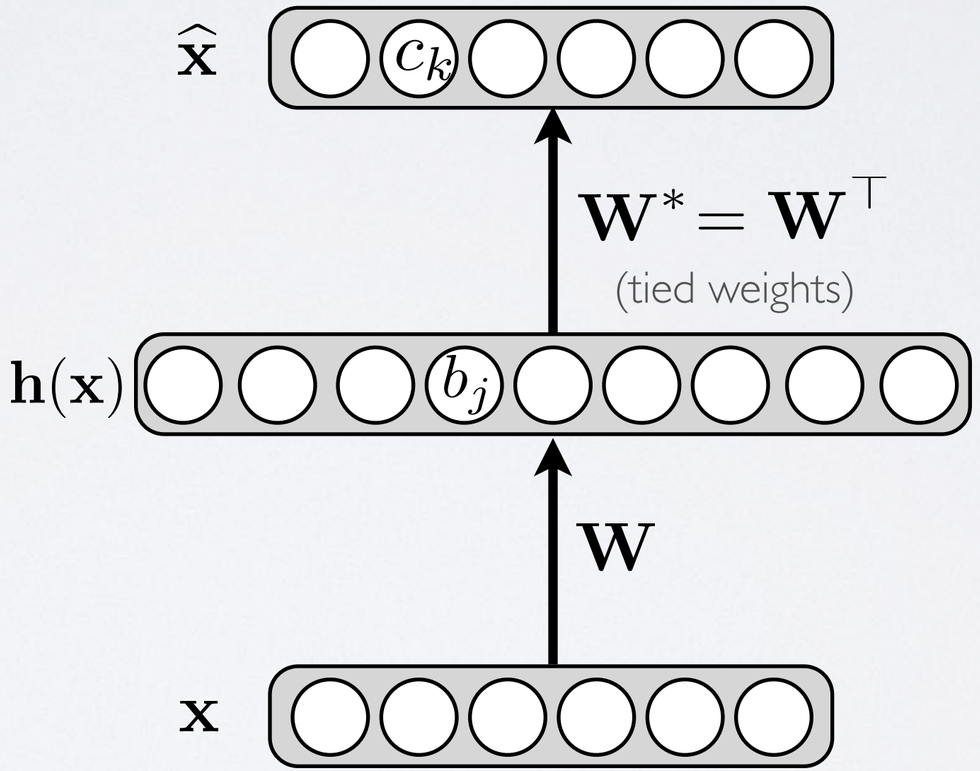




OVERCOMPLETE HIDDEN LAYER

Topics: overcomplete representation

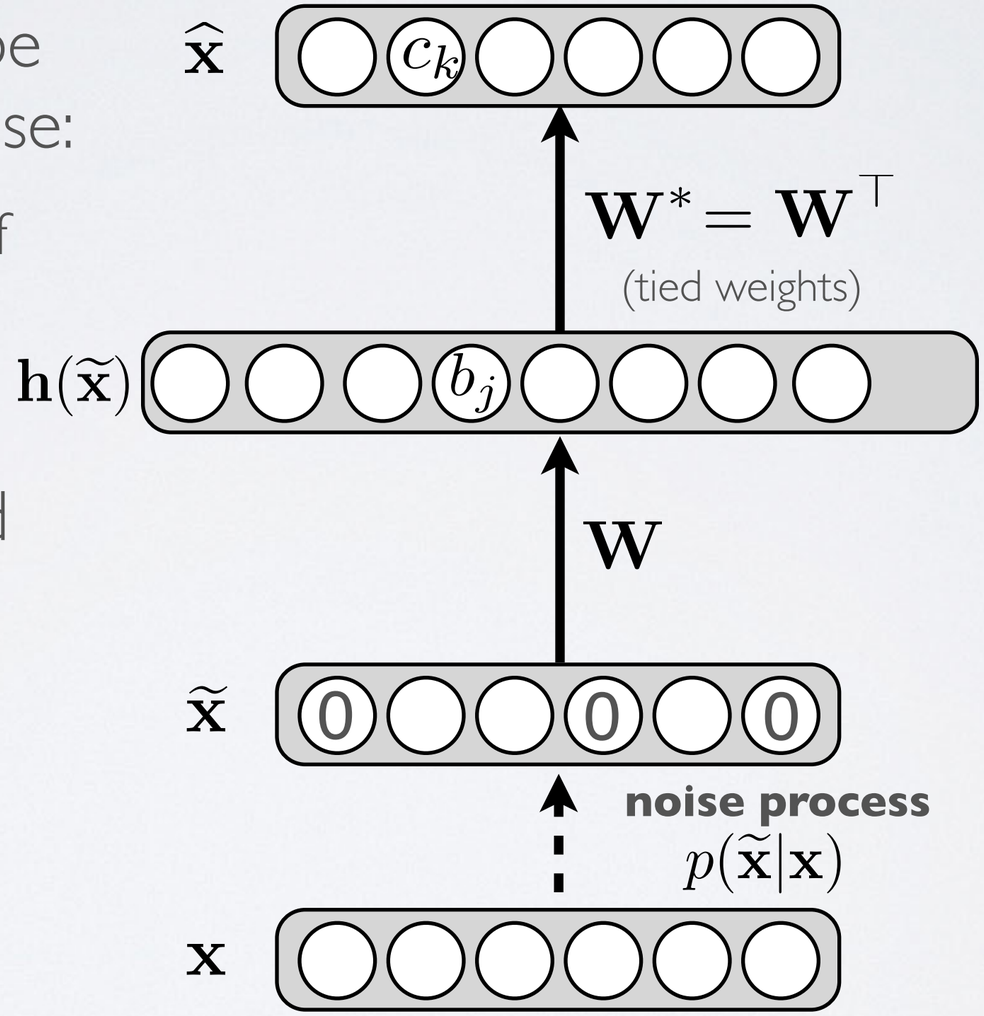
- Hidden layer is overcomplete if greater than the input layer
  - no compression in hidden layer
  - each hidden unit could copy a different input component
- No guarantee that the hidden units will extract meaningful structure



DENOISING AUTOENCODER

Topics: denoising autoencoder

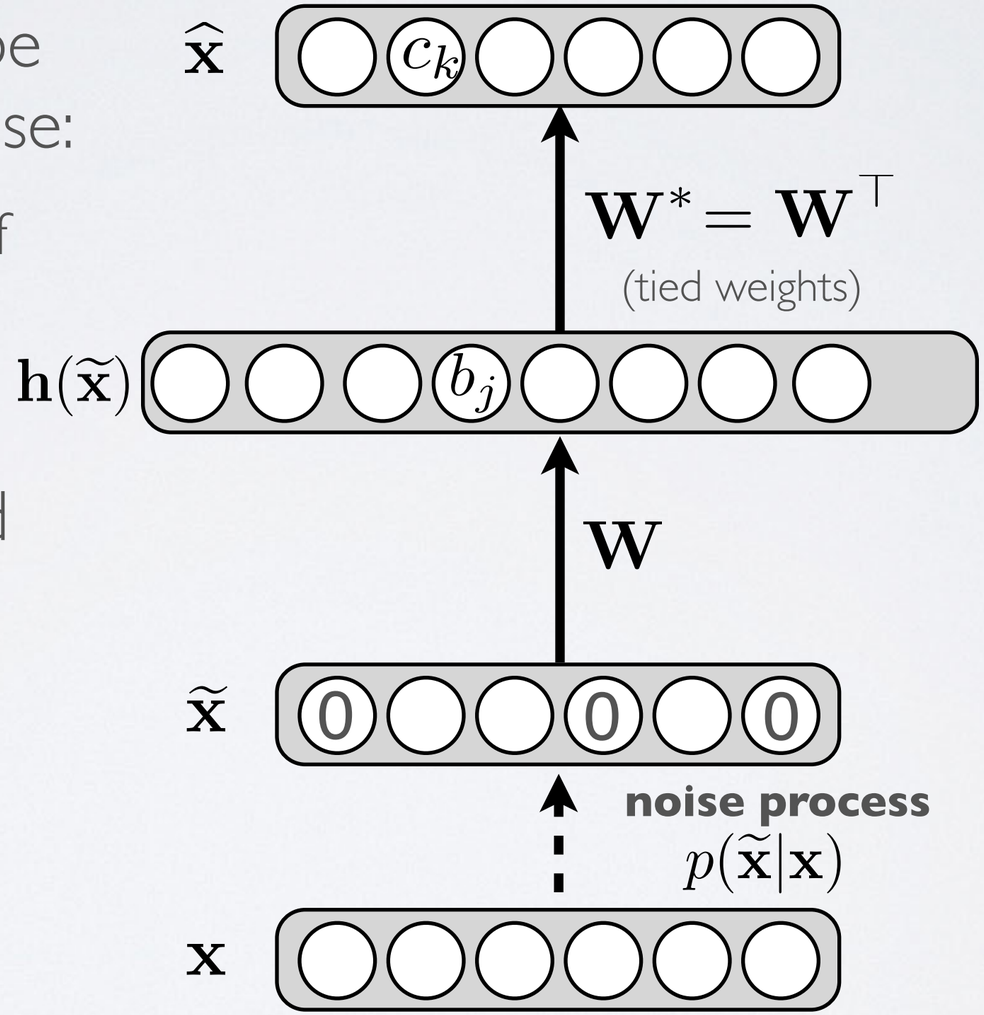
- Idea: representation should be robust to introduction of noise:
  - random assignment of subset of inputs to 0, with probability ν
  - Gaussian additive noise
- Reconstruction $\widehat{\mathbf{x}}$ computed from the corrupted input $\widetilde{\mathbf{x}}$
- Loss function compares $\widehat{\mathbf{x}}$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



DENOISING AUTOENCODER

Topics: denoising autoencoder

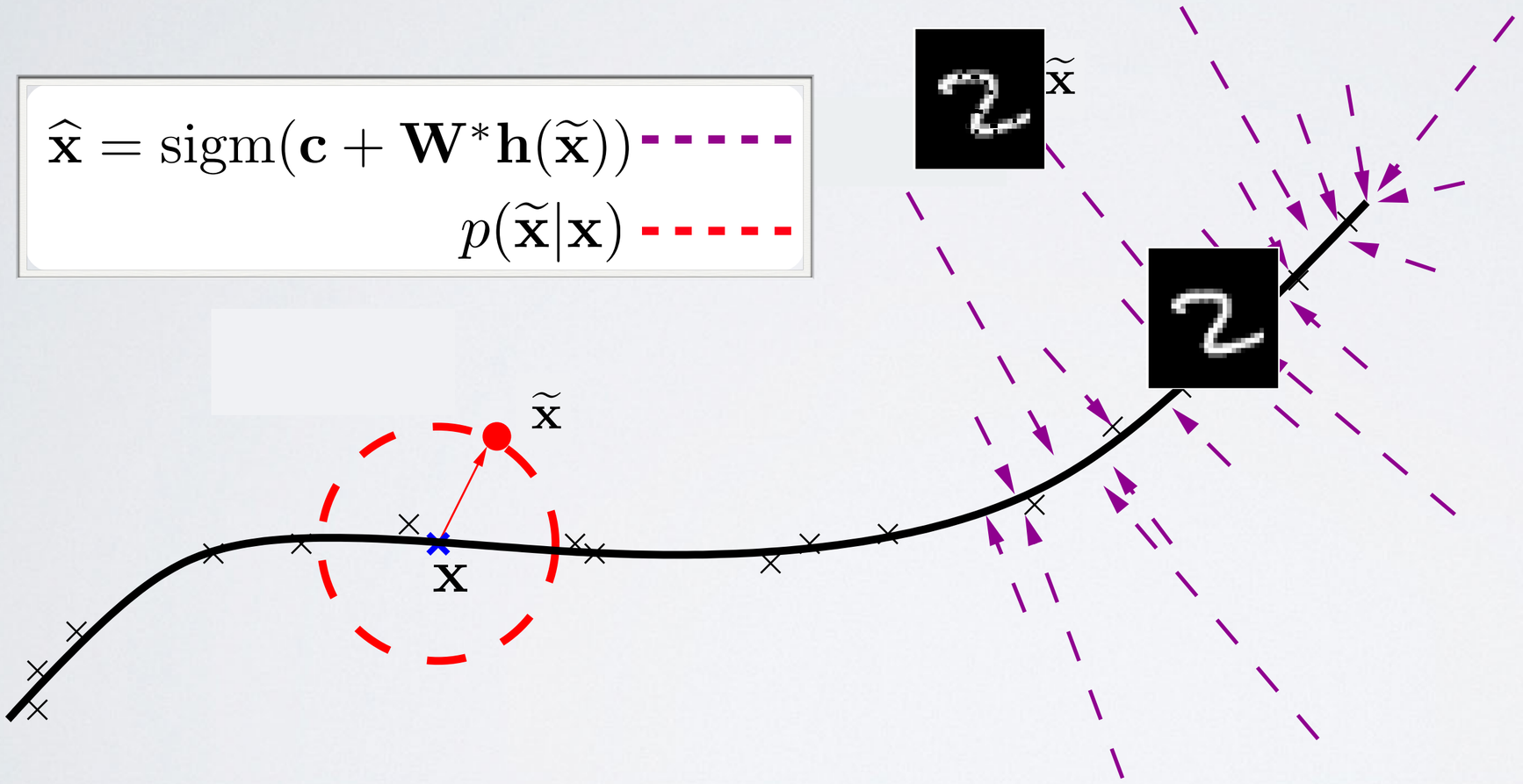
- Idea: representation should be robust to introduction of noise:
  - random assignment of subset of inputs to 0, with probability ν
  - Gaussian additive noise
- Reconstruction $\widehat{\mathbf{x}}$ computed from the corrupted input $\widetilde{\mathbf{x}}$
- Loss function compares $\widehat{\mathbf{x}}$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



DENOISING AUTOENCODER

$$\widehat{\mathbf{x}} = \text{sigm}(\mathbf{c} + \mathbf{W}^* \mathbf{h}(\widetilde{\mathbf{x}}))$$

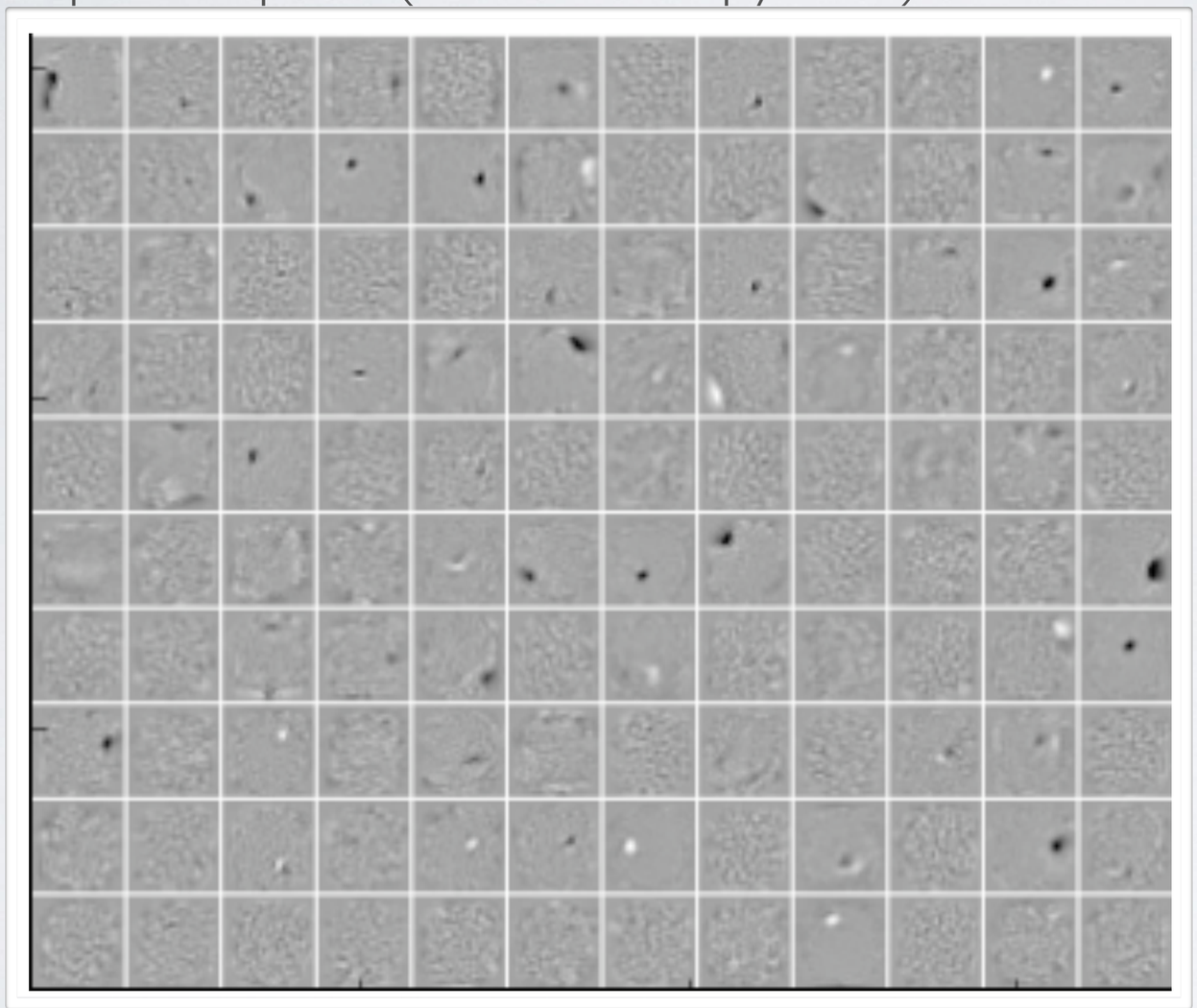
$$p(\widetilde{\mathbf{x}}|\mathbf{x})$$



FILTERS (DENOISING AUTOENCODER)

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

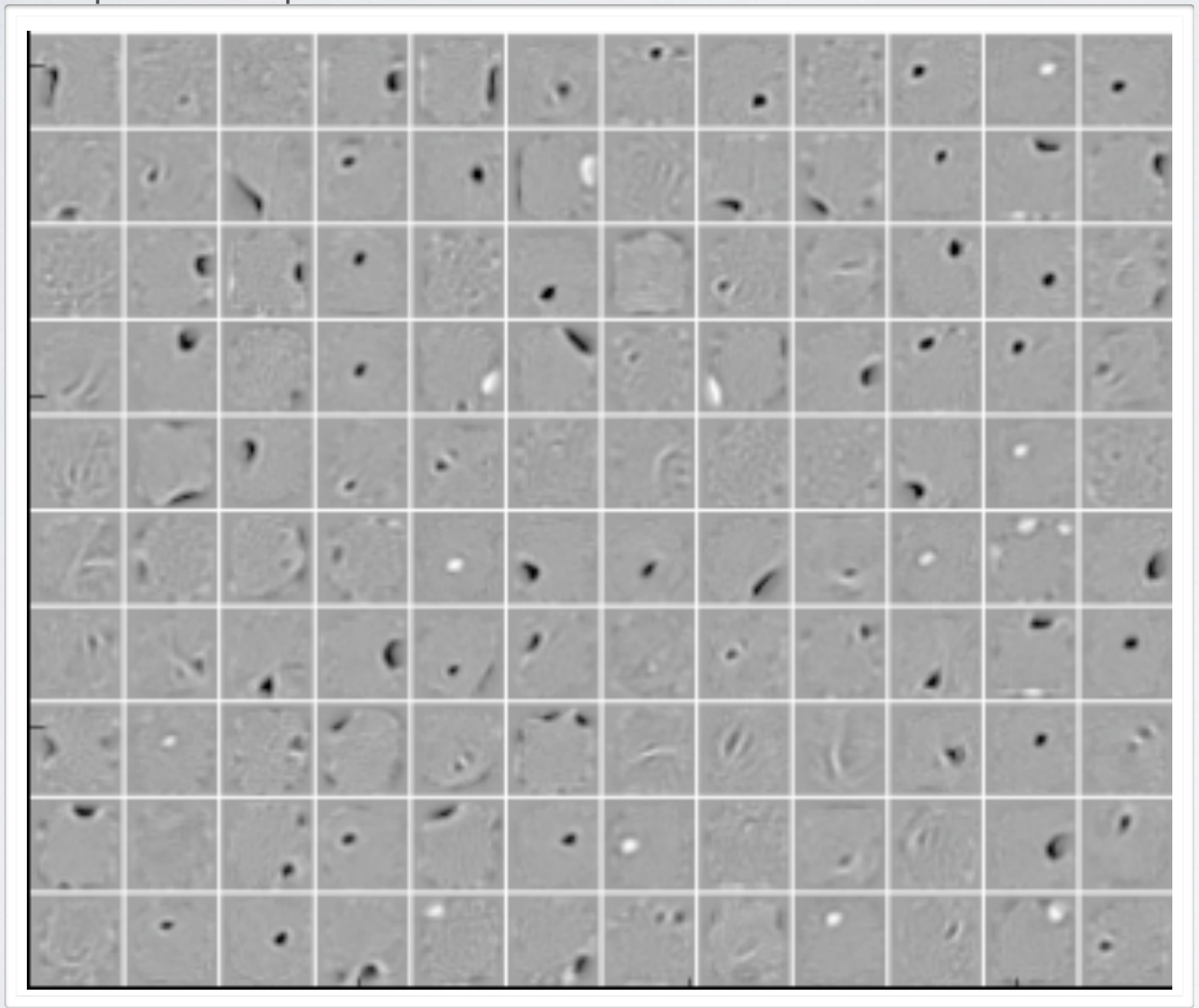
- No corrupted inputs (cross-entropy loss)



FILTERS (DENOISING AUTOENCODER)

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

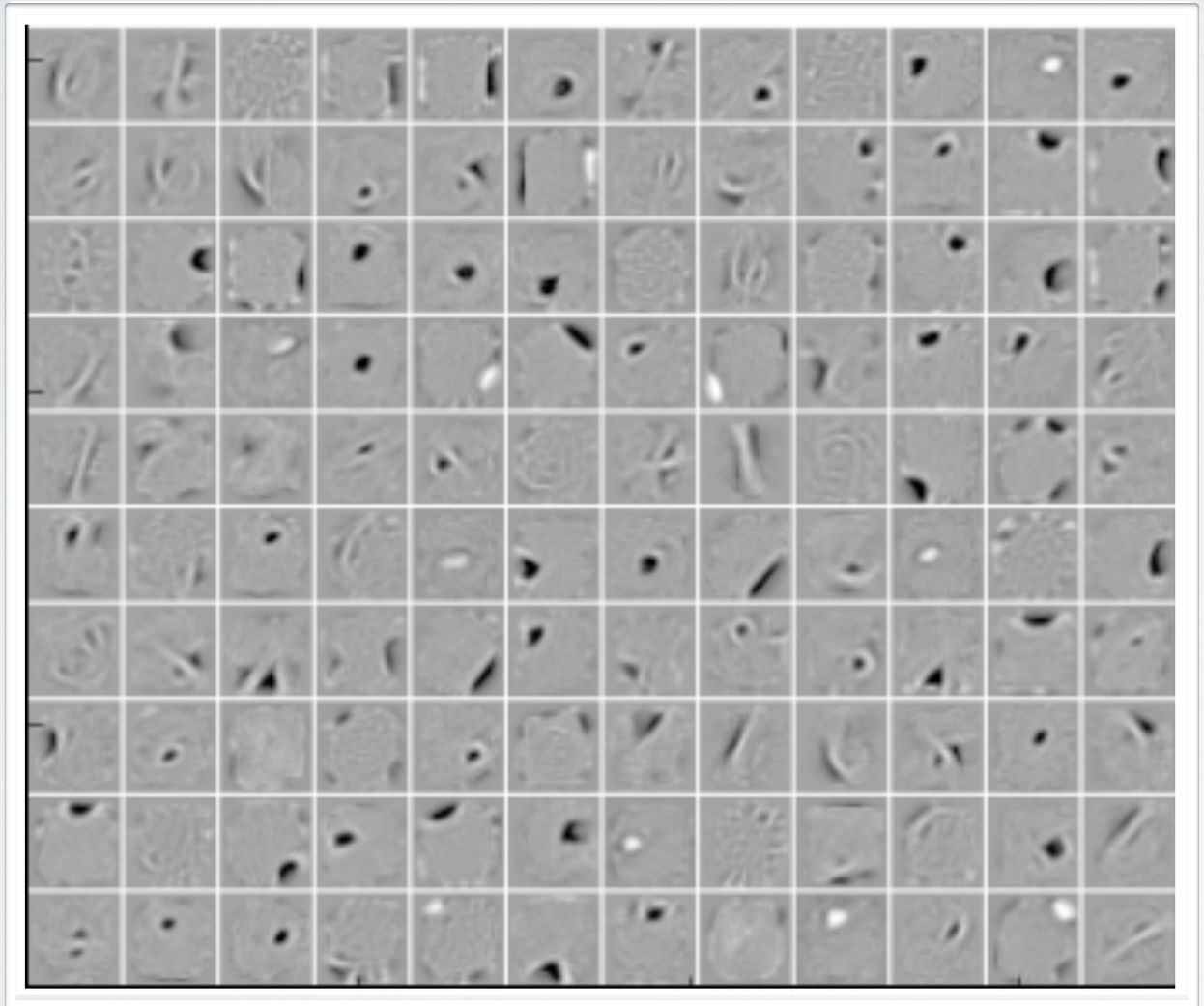
- 25% corrupted inputs



FILTERS (DENOISING AUTOENCODER)

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

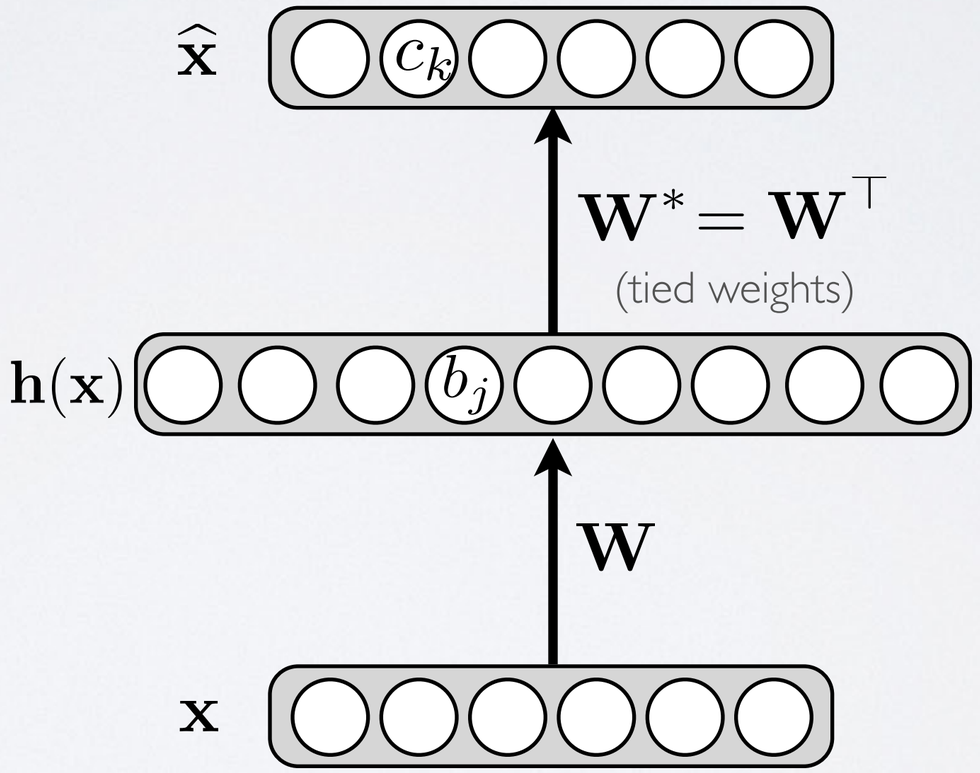
- 50% corrupted inputs



OVERCOMPLETE HIDDEN LAYER

Topics: overcomplete representation

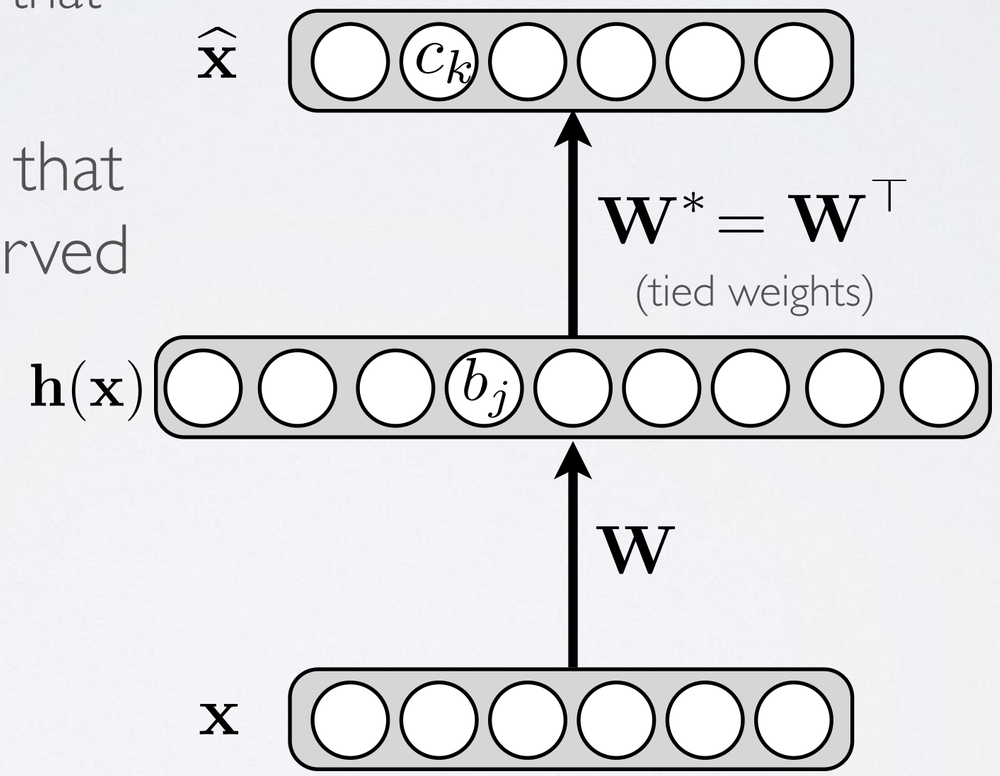
- Hidden layer is overcomplete if greater than the input layer
  - no compression in hidden layer
  - each hidden unit could copy a different input component
- No guarantee that the hidden units will extract meaningful structure



CONTRACTIVE AUTOENCODER

Topics: contractive autoencoder

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

Topics: contractive autoencoder

- New loss function:

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

  - where, for binary observations:

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

encoder keeps good information

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

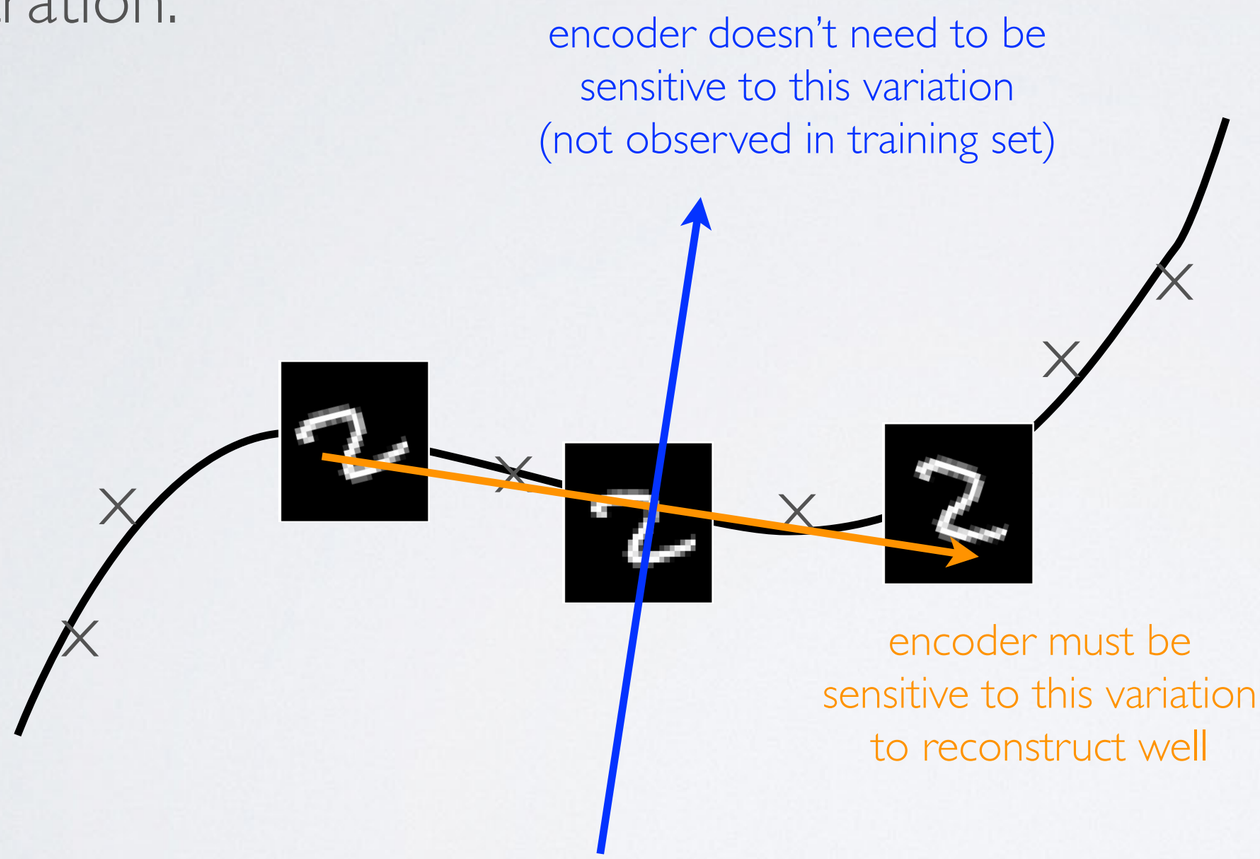
encoder throws away all information

encoder keeps **only** good information

CONTRACTIVE AUTOENCODER

Topics: contractive autoencoder

- Illustration:



WHICH AUTOENCODER ?

Topics: denoising autoencoder, contractive autoencoder

- Both the denoising and contractive autoencoder 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.