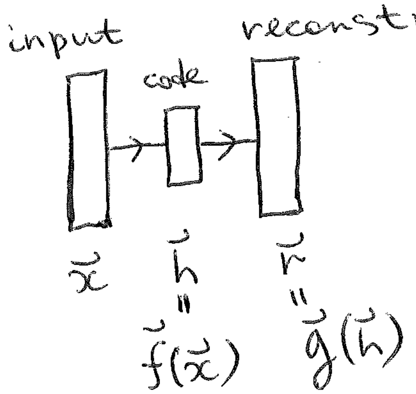
Lecture 21 Autoencoders



minimize loss function $\underbrace{L(\vec{x}, \vec{g}(\vec{f}(\vec{x})))}$

Learn $\vec{g}(\vec{f}(\vec{x})) = \vec{x}$

copies input to output

we're interested in the code $\vec{h}$: dimensionality reduction, feature learning

we need to avoid overfitting (i.e., just copying input to output):

use undercomplete autoencoder (code dim'n D_c < input dim'n D_i)

OR overcomplete autoencoder ($D_c > D_i$)

+ regularization for sparsity of the code, robustness to noise, robustness to translations / rotations, etc.

Sparse autoencoders: minimize

$$L(\vec{x}, \vec{g}(\vec{f}(\vec{x}))) + \lambda \Omega(\vec{h})$$

Could be used e.g. for feature learning which then provide input to classification. For example, one could use ReLU activation units

in the code layer $\oplus$ $\Omega(\vec{h}) = |\vec{h}|$
(i.e., to encode $\vec{h}$) L_1 penalty.

This should produce $\emptyset$'s in the $\vec{h}$ code.

<u>Denoising autoencoders</u>: minimize

$$L(\vec{x}, \vec{g}(\vec{f}(\vec{x}'))) \quad , \text{ where}$$

$$\vec{x}' = \vec{x} + \underbrace{\vec{\epsilon}}_{\text{noise}} \quad \leftarrow \text{perturbed input}$$

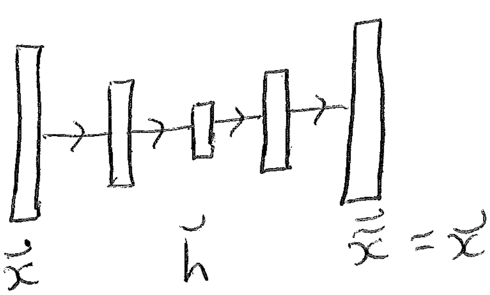
This forces autoencoders to undo the noise corruption.

<u>Contractive autoencoders</u>: minimize

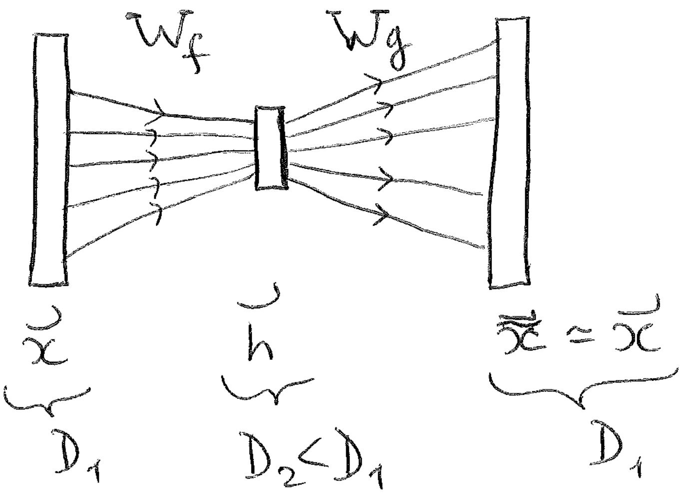
$$L(\vec{x}, \vec{g}(\vec{f}(\vec{x}))) + \lambda \underbrace{\sum_i \| \nabla_{\vec{x}} h_i \|^2}_{\text{''} \, \Omega(\vec{h}, \vec{x})}$$

$$h_i = f_i(\vec{x})$$

autoencoders → shallow (as discussed thus far)
autoencoders → deep (multiple layers to represent $\vec{f}$ & $\vec{g}$):



Ex.: linear autoencoder



$$\vec{h} = W_f \vec{x}$$
$$\tilde{\vec{x}} = W_g \vec{h}$$

$$\min_{W_f, W_g} \underbrace{\left\{ \frac{1}{2} \sum_n \left| W_g W_f \vec{x}_{\underset{\text{datapoint}}{n}} - \vec{x}_n \right|^2 \right\}}_{L \,=\, \text{loss function}}$$

Equiv. to PCA (principal component analysis)

non-linear autoencoder

$$\min_{W_f, W_g} \left\{ \frac{1}{2} \sum_n \left| \vec{g}(\vec{f}(\vec{x}_n ; W_f) ; W_g) - \vec{x}_n \right|^2 \right\}$$

NN weights + biases

[Projection onto lower-D non-linear manifold.]

Probabilistic autoencoders

f: $\Pr(\vec{h} \mid \vec{x}; W_f)$

g: $\Pr(\vec{x} \mid \vec{h}; W_g)$

use sigmoid, softmax or linear units at the code & output layers

to produce gaussian μ, σ

discard "−" if negative

$P(\tilde{\vec{x}} \mid \vec{h}; W_g)$

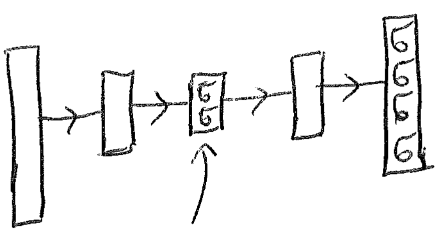
$\vec{x}, \tilde{\vec{x}}, \vec{h}$ are random vars

$P(\vec{h} \mid \vec{x}; W_f)$

sigmoid
softmax
unity (gaussian)

$P(\vec{x} \mid \vec{h}; W_g)$

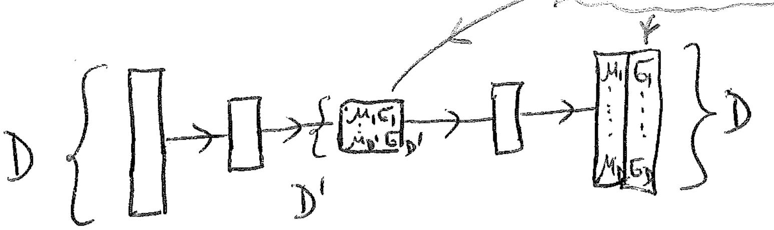
binary



$P(\vec{h} \mid \vec{x}; W_f)$

binary

no covariances



Can use these as generative models to produce novel outputs:

1. Sample $\vec{h}$ from $P(\vec{h})$
2. Sample $\vec{x}$ from decoder: $P(\vec{x} \mid \vec{h}; W_g)$

[Denoising autoencoders: (detailed discussion)]

1. Sample $\vec{x}$ from training data
2. Sample noisy/corrupted (version of $\vec{x}$):

$$\vec{x}' = \vec{x} + \vec{\epsilon} \Rightarrow P(\vec{x}'|\vec{x})$$

3. Use $(\vec{x}, \vec{x}')$ to construct

$$p_{decoder}(\vec{x}\,|\,\vec{h} = f(\vec{x}')) \equiv p_{reconstruct}(\vec{x}|\vec{x}')$$

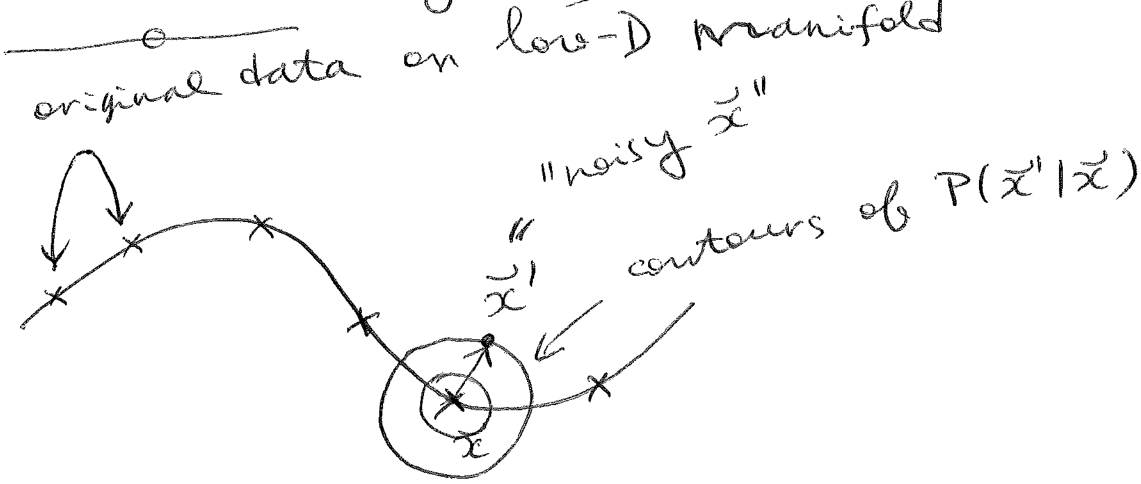
↳ minimize

$$-\log p_{decoder}(\vec{x}|\vec{h})$$

The error function is given by

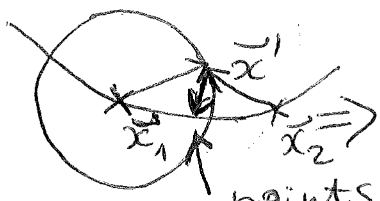
$$E = -\underbrace{\int d\vec{x} \int d\vec{x}'}\, p(\vec{x})\, P(\vec{x}'|\vec{x}) \log p_{decoder}(\vec{x}\,|\,\vec{h} = f(\vec{x}'))$$

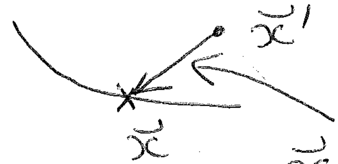
"summed" over many datapoints $\vec{x}$ &, for each $\vec{x}$, many corrupted/noisy $\vec{x}'$



Now, consider quadratic error:

$$E = \int d\vec{x} \int d\vec{x}' \, p(\vec{x}) \, P(\vec{x}'|\vec{x}) \, |\underbrace{\vec{g}(\vec{f}(\vec{x}'))}_{\text{learns to be as close to } \vec{x} \text{ as possible, by averaging over many } \vec{x}' \text{ realizations}} - \vec{x}|^2$$





points towards the nearest point on the manifold

averaged over many $\vec{x}$ (same $\vec{x}'$), →

$$\vec{x} - \vec{x}' \simeq \vec{g}(f(\vec{x}')) - \vec{x}'$$

represents a gradient flow towards the "true" $\vec{x}$-manifold

requires local linearity of the "true" manifold