Advanced Machine Learning
Summer 2019

Part 18 – Variational Autoencoders
03.07.2019

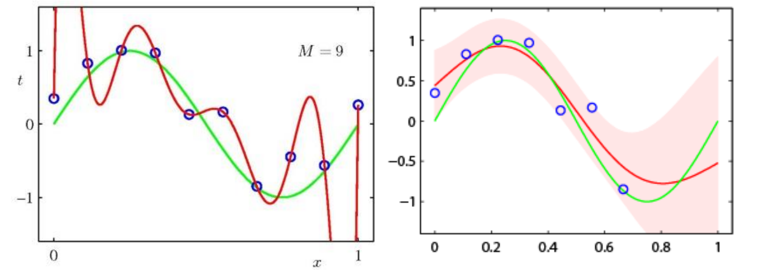
Prof. Dr. Bastian Leibe

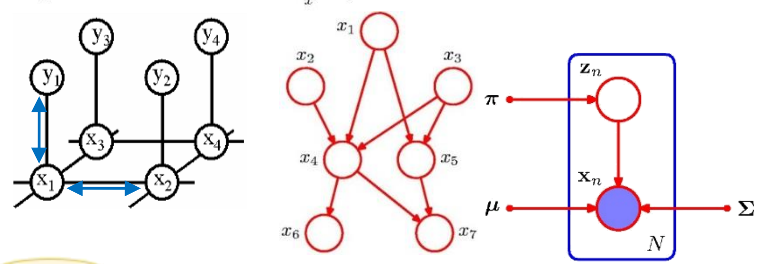
RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de

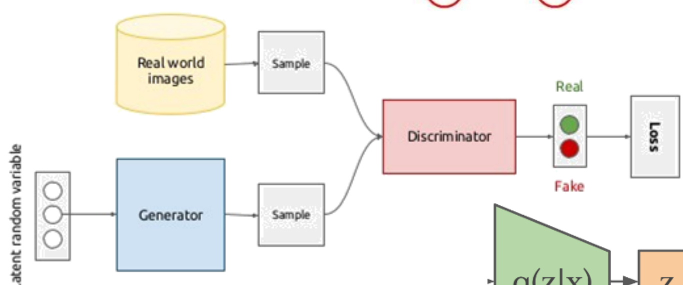
Course Outline

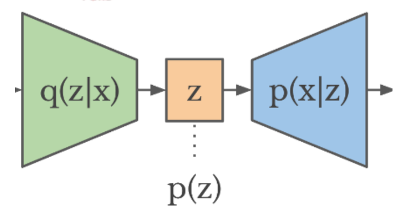
- Regression Techniques
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
- Deep Reinforcement Learning
- Probabilistic Graphical Models
  - Bayesian Networks
  - Markov Random Fields
  - Inference (exact & approximate)
  - Latent Variable Models
- Deep Generative Models
  - Generative Adversarial Networks
  - Variational Autoencoders

$$f : \mathcal{X} \rightarrow \mathbb{R}$$







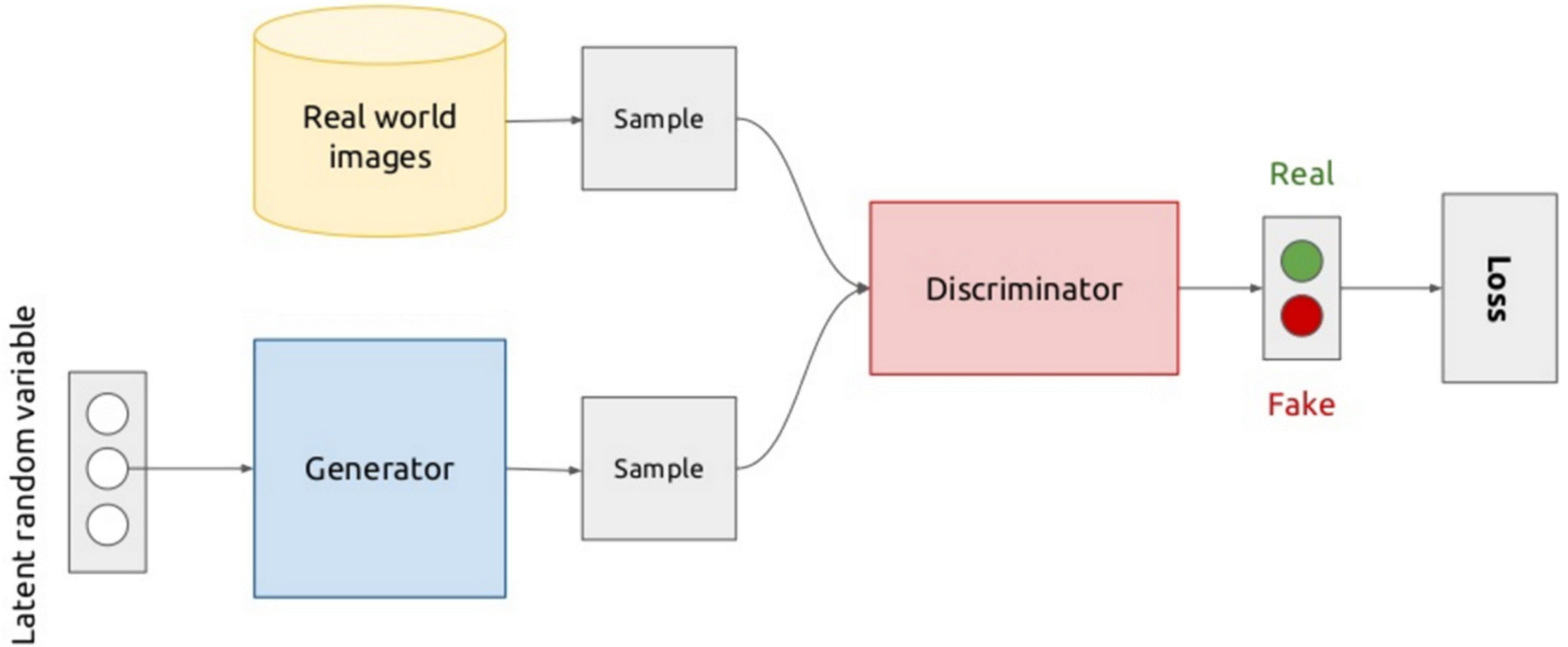


Topics of This Lecture

- Recap: GANs
- Autoencoders
  - Motivation
  - Regularized Autoencoder
  - Denoising Autoencoder
- Variational Autoencoders (VAE)
  - Autoencoders as Generative Models
  - Intractability
  - Variational Approximation
  - Evidence Lower Bound (ELBO)
- Application Examples

Recap: Generative Adversarial Networks (GANs)

- Conceptual view



- Main idea
  - Simultaneously train an image generator G and a discriminator D.
  - Interpreted as a two-player game

Image credit: Kevin McGuiness

Recap: GAN Loss Function

- This corresponds to a two-player minimax game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[\log\left(1 - D\left(G(\boldsymbol{z})\right)\right)\right]$$

- Explanation
  - Train D to maximize the probability of assigning the correct label to both training examples and samples from G.
  - Simultaneously train G to minimize $\log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$.

- The Nash equilibrium of this game is achieved at
  - $p_g(\boldsymbol{x}) = p_{data}(\boldsymbol{x}) \quad \forall \boldsymbol{x}$
  - $D(\boldsymbol{x}) = \frac{1}{2} \quad \forall \boldsymbol{x}$

GAN Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

Discriminator updates

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

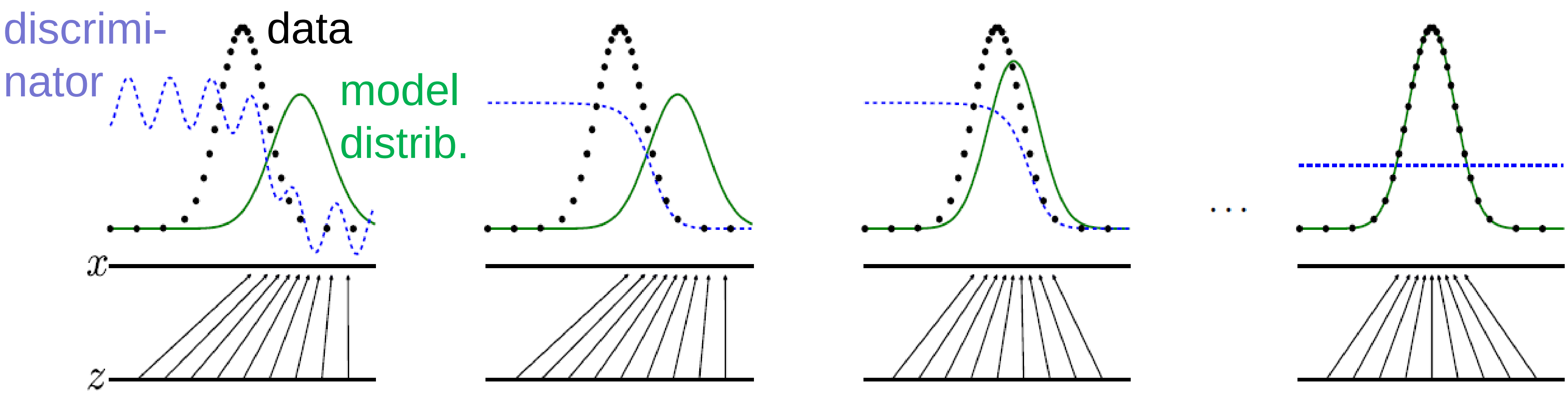
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

Generator updates

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Recap: Intuition

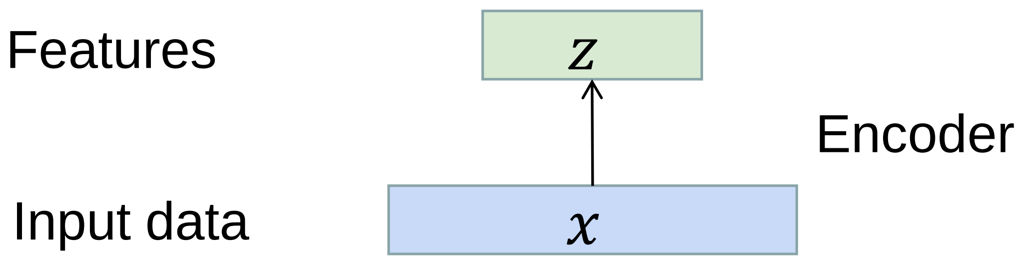


- Behavior near convergence
  - In the inner loop, D is trained to discriminate samples from data.
  - Gradient of D guides G to flow to regions that are more likely to be classified as data.
  - After several steps of training, G and D will reach a point at which they cannot further improve, because $p_g = p_{data}$.
  - Now, the discriminator is unable to differentiate between the two distributions, i.e., $D(\boldsymbol{x}) = 0.5$.

Topics of This Lecture

- Recap: GANs
- Autoencoders
  - Motivation
  - Regularized Autoencoder
  - Denoising Autoencoder
- Variational Autoencoders (VAE)
  - Autoencoders as Generative Models
  - Intractability
  - Variational Approximation
  - Evidence Lower Bound (ELBO)
- Application Examples

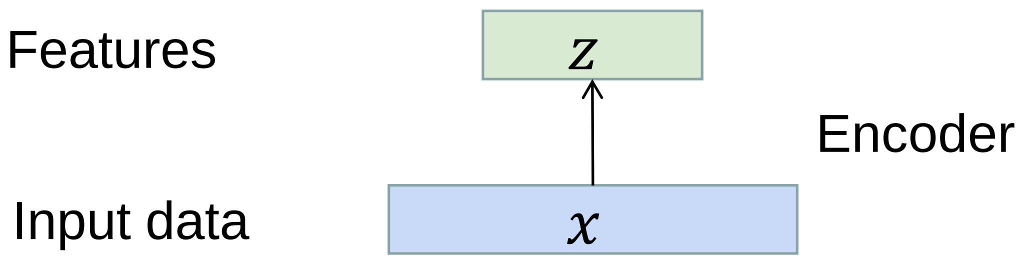
Autoencoders



- Autoencoders
  - Unsupervised learning approach for learning a lower-dimensional feature representation $\mathbf{z}$ from unlabeled input data $\mathbf{x}$.
  - $\mathbf{z}$ usually smaller than $\mathbf{x}$ (dimensionality reduction)
  - Want to capture meaningful factors of variation in the data

Slide credit: Feifei Li

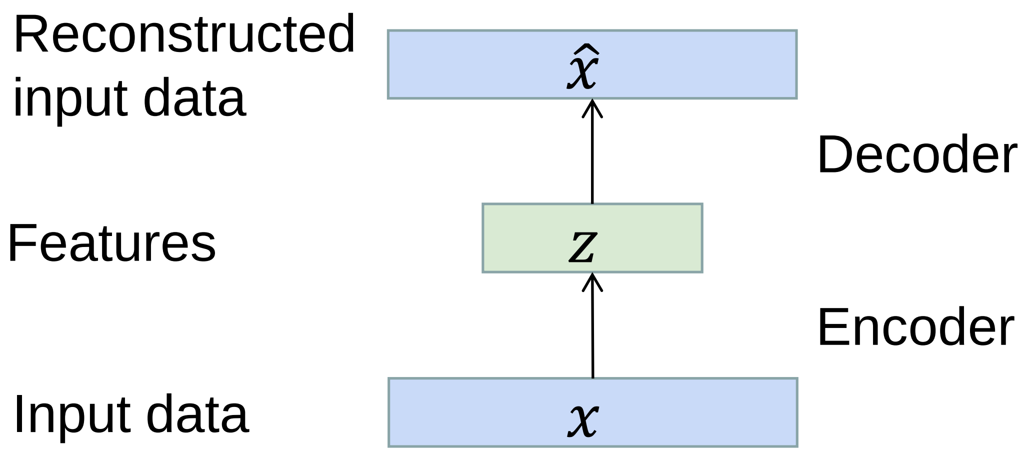
Autoencoders



- Encoder
  - Originally: shallow function (linear + sigmoid)
  - Later: Deep, fully-connected
  - Later: ReLU CNN

Slide credit: Feifei Li

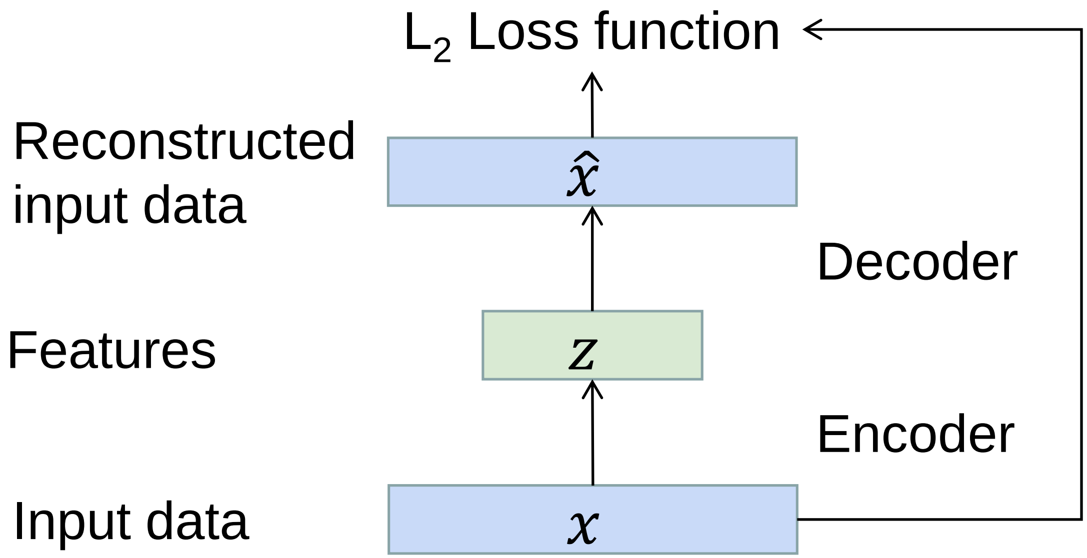
Autoencoders

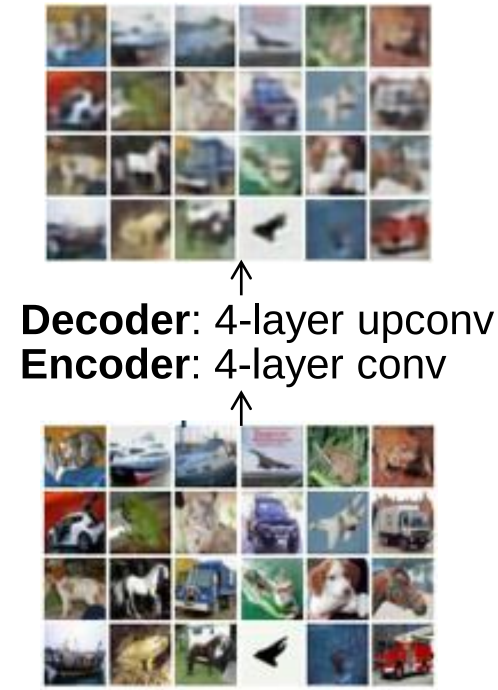


- How to learn such a feature representation?
  - Train such that features can be used to reconstruct original data.
  - "Autoencoding" – encoding itself

Slide credit: Feifei Li

Autoencoders

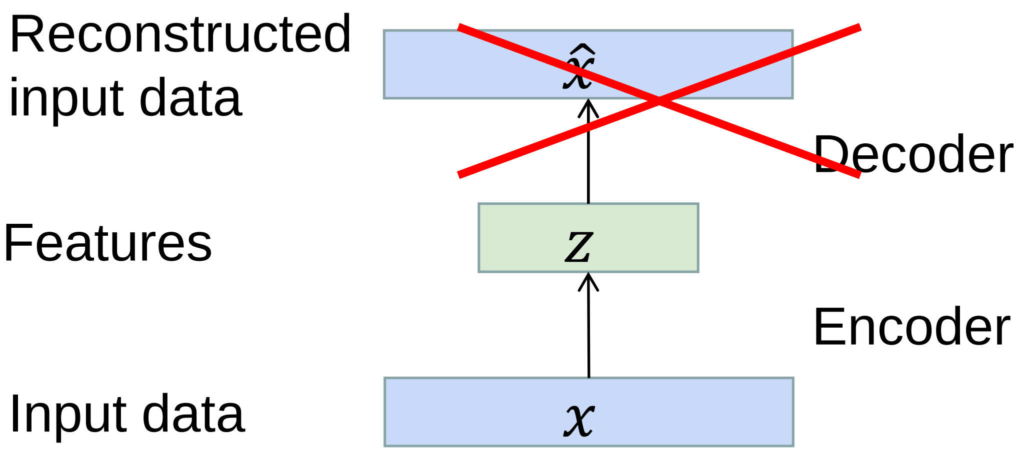




- How to learn such a feature representation?
  - Train such that features can be used to reconstruct original data.
  - "Autoencoding" – encoding itself
  - L_2 loss function $||x - \hat{x}||^2$
  - Note: this doesn't use any labels!

Slide credit: Feifei Li

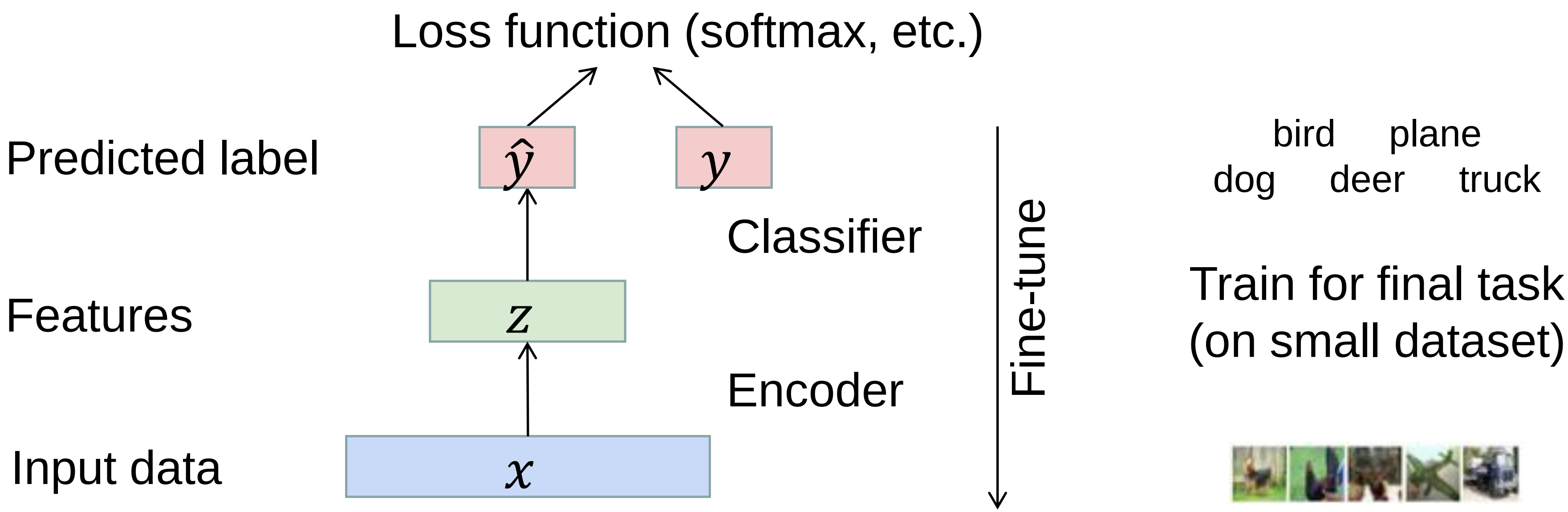
Autoencoders



- After training
  - Throw away the decoder part

Slide credit: Feifei Li

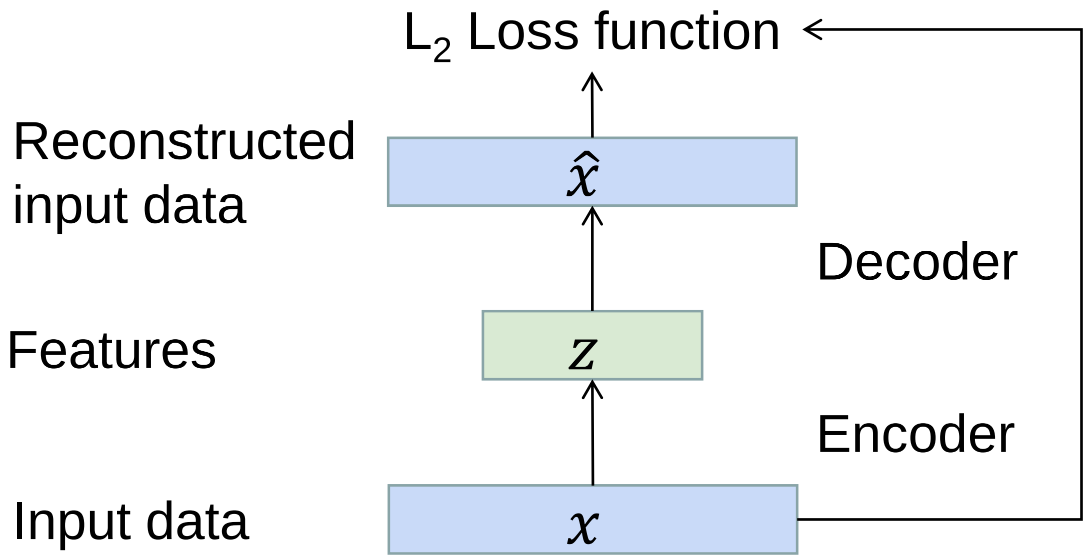
Autoencoders



- After training
  - Throw away the decoder part
  - Encoder can be used to initialize a supervised model
  - Fine-tune encoder jointly with supervised model
  - *Idea used in the 90s and early 2000s to pre-train deeper models*

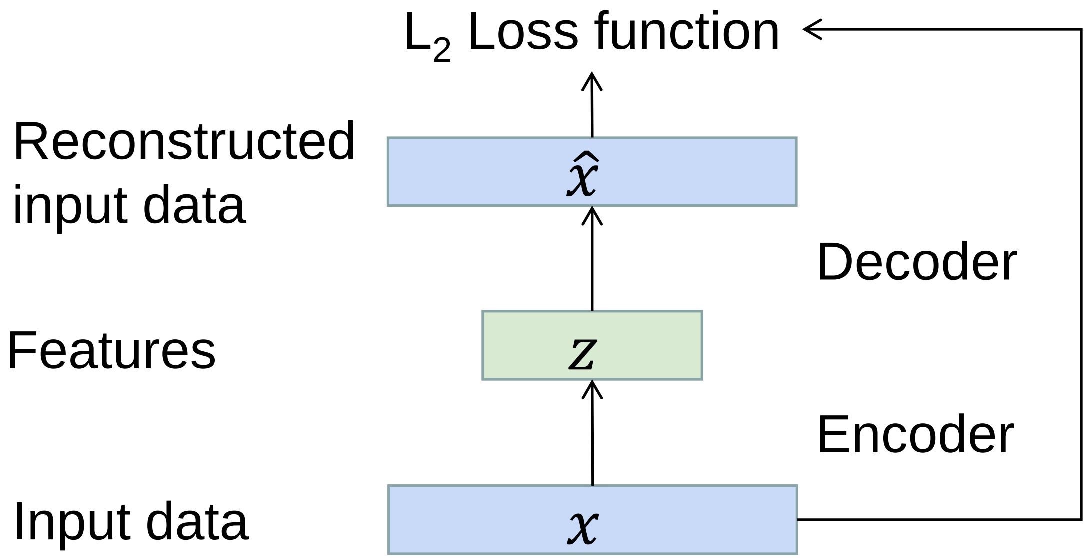
Slide credit: Feifei Li

Variants of Autoencoders



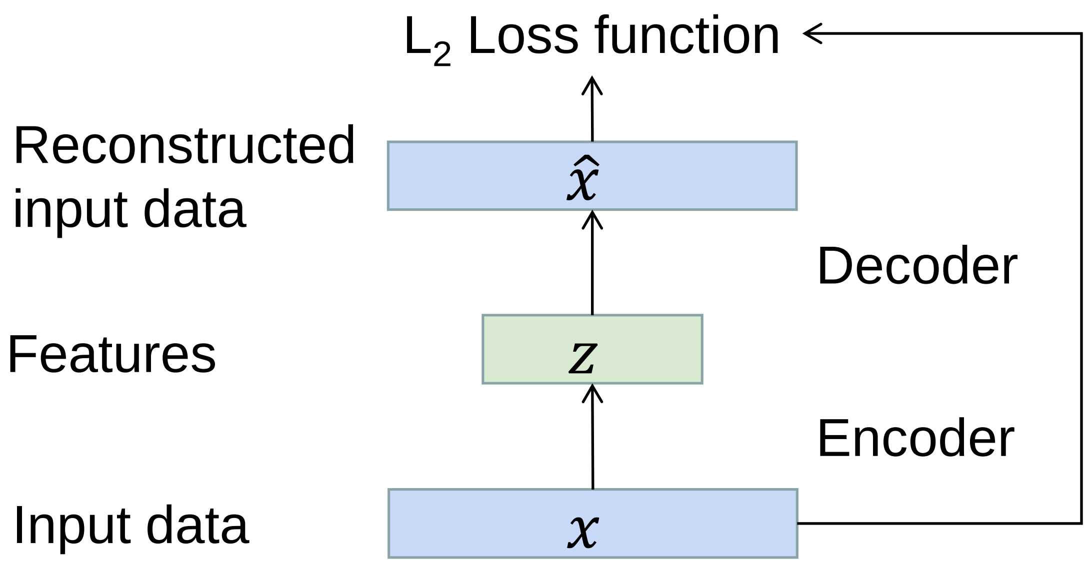
- Analyzing the learning process
  - Learning process minimizes a loss function $L\left(\mathbf{x}, g\left(f\left(\mathbf{x}\right)\right)\right)$
  - Linear decoder + L_2 loss: Autoencoder learns PCA subspace
  - Autoencoders with nonlinear encoder and decoder functions thus learn a more powerful nonlinear generalization of PCA.

Variants of Autoencoders



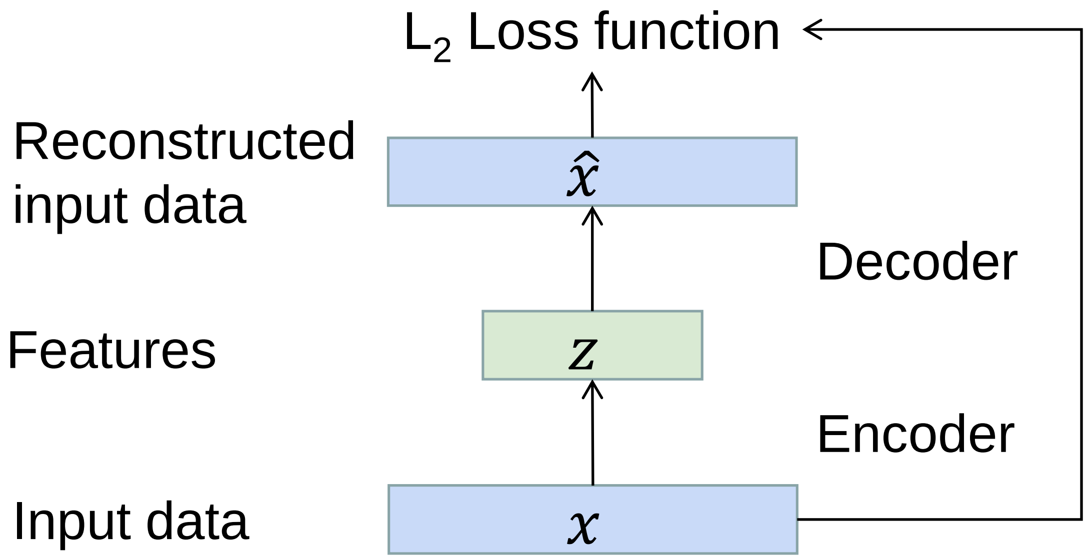
- Analyzing the learning process
  - Learning process minimizes a loss function $L\left(\mathbf{x}, g\left(f\left(\mathbf{x}\right)\right)\right)$
  - *Unfortunately, if the encoder and decoder are too powerful, they can learn to perform the copying task without learning useful information.*
  - E.g., learn a 1D code to memorize each training example.

Variants of Autoencoders



- Regularized Autoencoders
  - Include a regularization term to the loss function: $L\left(\mathbf{x}, g\left(f(\mathbf{x})\right)\right) + \Omega(\mathbf{z})$
  - E.g., enforce sparsity by an L_1 regularizer $\Omega(\mathbf{z}) = \lambda \sum_i |z_i|$

Variants of Autoencoders

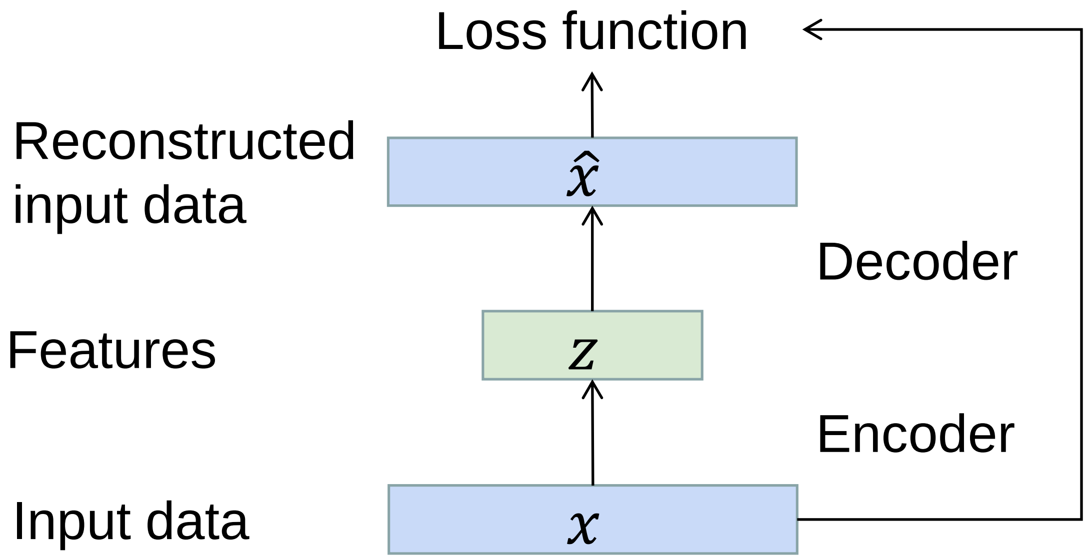


- Regularized Autoencoders
  - We can think of the sparse encoder framework as approximating ML training of a generative model with latent variables $\mathbf{z}$.

$$\log p_{model}(\mathbf{x}) = \log \sum_{\mathbf{z}} p_{model}(\mathbf{x}, \mathbf{z})$$

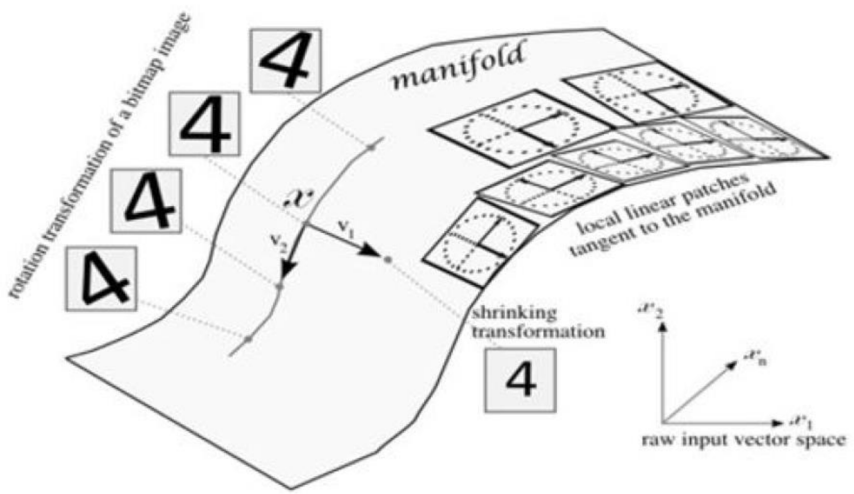
  - The autoencoder approximates this sum with a point estimate for just one highly likely value for $\mathbf{z}$.

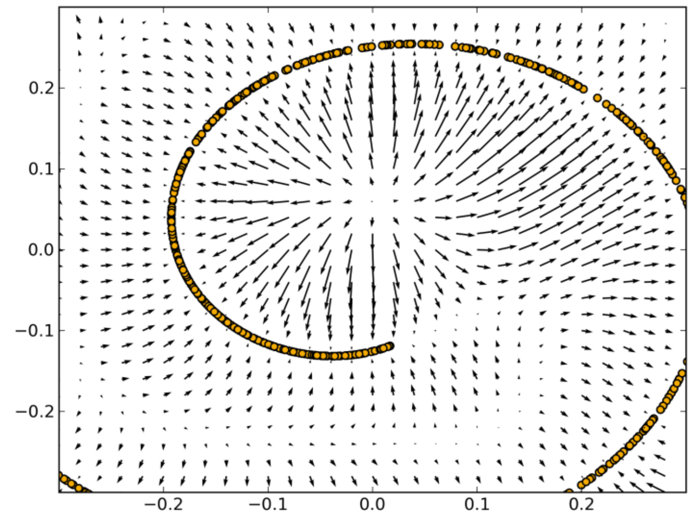
Variants of Autoencoders



- Denoising Autoencoder (DAE)
  - Rather than the reconstruction loss, minimize $L\left(\mathbf{x}, g\left(f(\tilde{\mathbf{x}})\right)\right)$ where $\tilde{\mathbf{x}}$ is a copy of $\mathbf{x}$ that has been corrupted by some noise.
  - Denoising forces f and g to implicitly learn the structure of $p_{data}(\mathbf{x})$.

Variants of Autoencoders





- Denoising Autoencoder (DAE)
  - Assumption: Natural data actually lies in a (low-dimensional) manifold of the high-dimensional space of input data $\mathbf{x}$.
  - By corrupting the input data with noise, we force the DAE to learn a vector field that pushes towards this low-dimensional manifold.

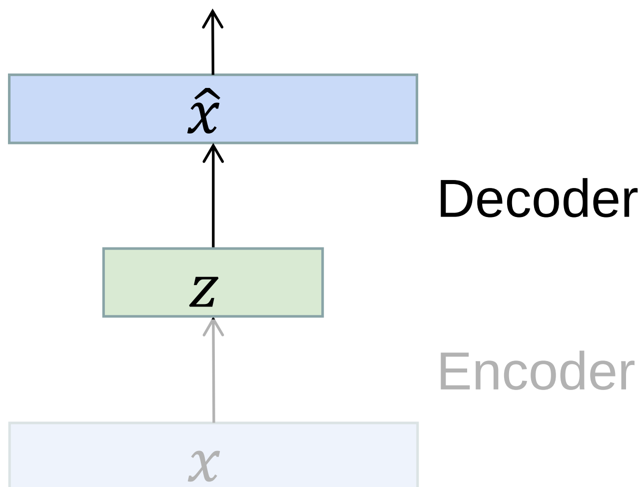
Image source: [Goodfellow 2016]

Topics of This Lecture

- Recap: GANs
- Autoencoders
  - Motivation
  - Regularized Autoencoder
  - Denoising Autoencoder
- Variational Autoencoders (VAE)
  - Autoencoders as Generative Models
  - Intractability
  - Variational Approximation
  - Evidence Lower Bound (ELBO)
- Application Examples

Autoencoders as Data Generators

- Autoencoders
  - Can reconstruct data and can learn features to initialize a supervised model
  - Features capture factors of variation in training data
  - Can we generate new images from an autoencoder?



  - For this we need to generate samples from the data manifold. How?

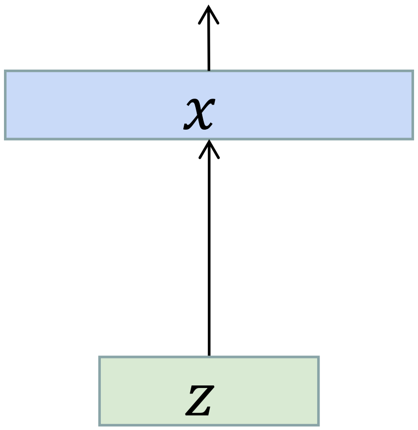
Slide inspired by Feifei Li

Probabilistic Spin on Autoencoders

- Idea: Sample the model to generate data
  - Assume training data $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ is generated from underlying latent representation $\mathbf{z}$.

Sample from true conditional $p_{\theta^*}(\mathbf{x}|\mathbf{z}^{(i)})$

Sample from true prior $p_{\theta^*}(\mathbf{z})$



Slide credit: Feifei Li

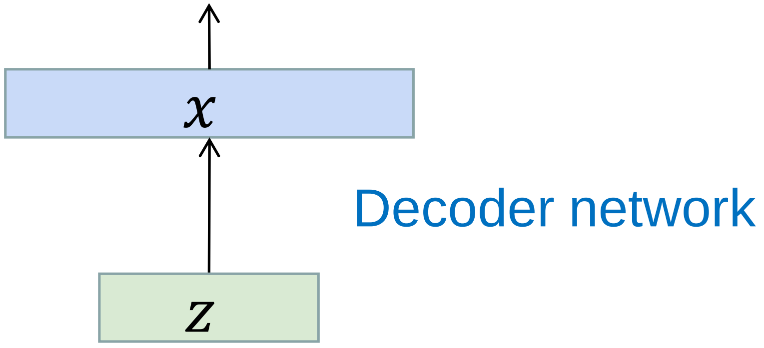
Probabilistic Spin on Autoencoders

Sample from true conditional

$$p_{\theta^*}(\mathbf{x}|\mathbf{z}^{(i)})$$

Sample from true prior

$$p_{\theta^*}(\mathbf{z})$$



- Idea: Sample the model to generate data
  - We want to estimate the true parameters θ^* of this generative model.
- How should we represent the model?
  - Choose prior $p(\mathbf{z})$ to be simple, e.g., Gaussian
  - Conditional $p(\mathbf{x} \mid \mathbf{z})$ is complex (generates image)

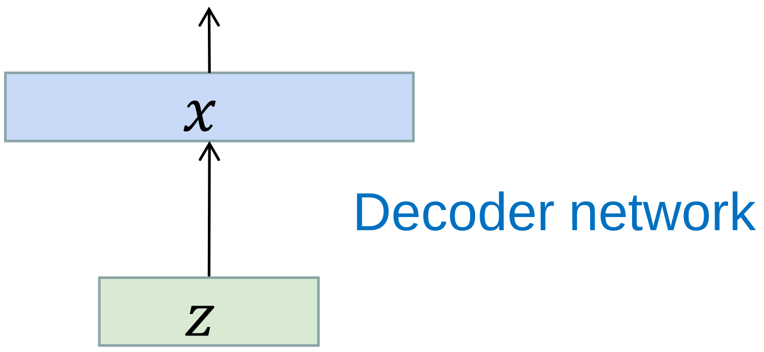
  $\Rightarrow$ Represent with neural network

Slide credit: Feifei Li

Probabilistic Spin on Autoencoders

Sample from true conditional
$p_{\theta^*}(\mathbf{x}|\mathbf{z}^{(i)})$

Sample from true prior
$p_{\theta^*}(\mathbf{z})$



- Idea: Sample the model to generate data
  - We want to estimate the true parameters θ^* of this generative model.
- How to train the model?
  - Learn model parameters to maximize likelihood of training data

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) d\mathbf{z}$$

  - *What is the problem here?* Intractable!

Slide credit: Feifei Li

Variational Autoencoders: Intractibility

- Computing the data likelihood

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) d\mathbf{z}$$

  - $p(\mathbf{z})$ is a simple Gaussian prior. $\Rightarrow$ ok.
  - $p(\mathbf{x} \mid \mathbf{z})$ is a decoder Neural network. $\Rightarrow$ ok.
  - But is is intractable to compute $p(\mathbf{x} \mid \mathbf{z})$ for every $\mathbf{z}$!

  - Posterior density is also intractable

$$p_\theta(\mathbf{z} \mid \mathbf{x}) = \frac{p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z})}{p_\theta(\mathbf{x})}$$

Slide credit: Feifei Li

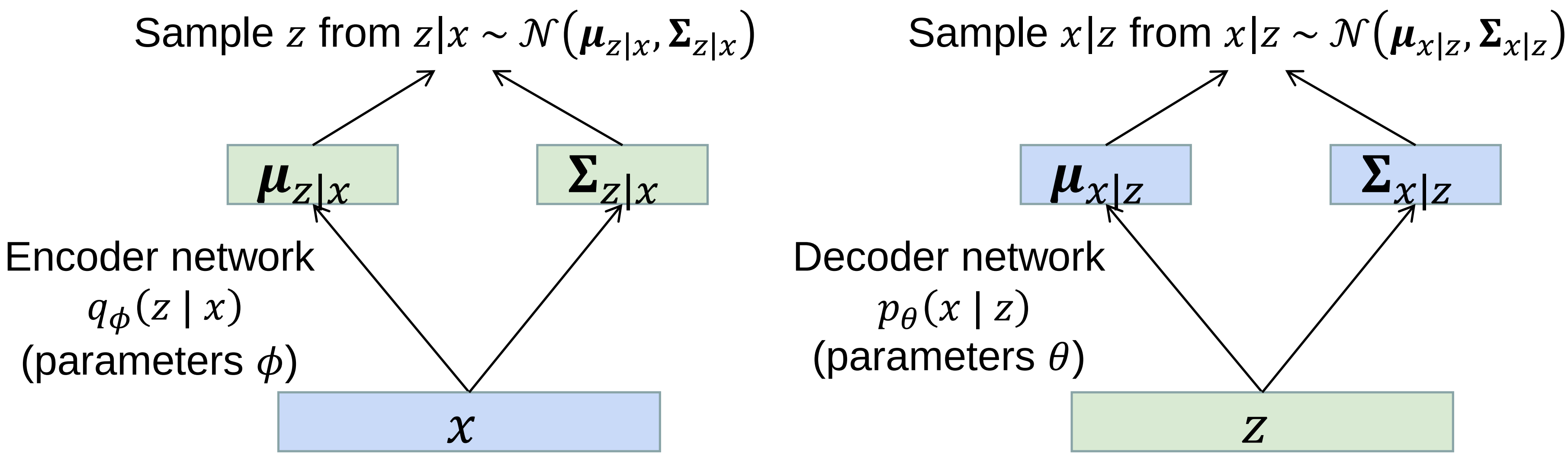
Variational Autoencoders: Intractibility

- Solution
  - In addition to the decoder network modeling $p_\theta(\mathbf{x} \mid \mathbf{z})$, define additional encoder network modeling $q_\phi(\mathbf{z} \mid \mathbf{x})$ that approximates $p_\theta(\mathbf{z} \mid \mathbf{x})$.
  - *We will see that this allows us to derive a lower bound on the data likelihood that is tractable and that we can optimize.*

Slide credit: Feifei Li

Variational Autoencoders

– Since we are modelling probabilistic generation of data, encoder and decoder networks are probabilistic



– Encoder and decoder networks are also called recognition/inference and generation networks

Slide credit: Feifei Li

Variational Autoencoders

- We can now work out the log-likelihood

$$\log p_\theta\left(x^{(i)}\right) = \mathbb{E}_{z \sim q_\phi\left(z \middle| x^{(i)}\right)}\left[\log p_\theta\left(x^{(i)}\right)\right]$$

($p_\theta\left(x^{(i)}\right)$ does not depend on z)

Taking expectation w.r.t. z (using encoder network) will come in handy later

Slide credit: Feifei Li

Variational Autoencoders

- We can now work out the log-likelihood

$$\log p_\theta(x^{(i)}) = \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ does not depend on } z)$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbb{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right]$$

$$= \mathbb{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}\left(q_\phi(z \mid x^{(i)}) \| p_\theta(z)\right) + D_{KL}\left(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)})\right)$$

The expectation w.r.t z
(using encoder network) lets
us write nice KL terms

Slide credit: Feifei Li

Variational Autoencoders

- We can now work out the log-likelihood

$$\log p_\theta(x^{(i)}) = \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ does not depend on } z)$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbb{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right]$$

$$= \mathbb{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)}))$$

Decoder network gives $p_\theta(\mathbf{x} \mid \mathbf{z})$, can compute estimate of this term through sampling.
(Sampling differentiable through reparametrization trick, see paper)

Slide credit: Feifei Li

Variational Autoencoders

- We can now work out the log-likelihood

$$\log p_\theta(x^{(i)}) = \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ does not depend on } z)$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbb{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right]$$

$$= \mathbb{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)}))$$

This KL term (between Gaussians for encoder/prior) has a nice closed-form solution

Slide credit: Feifei Li

Variational Autoencoders

- We can now work out the log-likelihood

$$\log p_\theta\left(x^{(i)}\right) = \mathbb{E}_{z\sim q_\phi\left(z|x^{(i)}\right)}\left[\log p_\theta\left(x^{(i)}\right)\right] \quad (p_\theta(x^{(i)}) \text{ does not depend on } z)$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta\left(x^{(i)} \mid z\right)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbb{E}_z\left[\log \frac{p_\theta\left(x^{(i)} \mid z\right)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi\left(z \mid x^{(i)}\right)}{q_\phi\left(z \mid x^{(i)}\right)}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbb{E}_z\left[\log p_\theta\left(x^{(i)} \mid z\right)\right] - \mathbb{E}_z\left[\log \frac{q_\phi\left(z \mid x^{(i)}\right)}{p_\theta(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi\left(z \mid x^{(i)}\right)}{p_\theta\left(z \mid x^{(i)}\right)}\right]$$

$$= \mathbb{E}_z\left[\log p_\theta\left(x^{(i)} \mid z\right)\right] - D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \| p_\theta(z)\right) + D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \| p_\theta\left(z \mid x^{(i)}\right)\right)$$

$p_\theta(\mathbf{z} \mid \mathbf{x})$ intractable (as seen earlier), can't compute this KL term ☹
But we know KL divergence always ≥ 0.

Slide credit: Feifei Li

Variational Autoencoders

- We can now work out the log-likelihood

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ does not depend on } z)$$

Want to maximize data likelihood

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right]$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}\left(q_\phi(z \mid x^{(i)}) \| p_\theta(z)\right)}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}\left(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)})\right)}_{\geq 0}$$

Tractable lower bound, which we can take gradient of and optimize

Slide credit: Feifei Li

Variational Autoencoders

- Variational Lower Bound ("ELBO")

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

$$= \underbrace{\mathbb{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right]}_{\text{"Reconstruct the input data"}} - \underbrace{D_{KL}\left(q_\phi(z \mid x^{(i)}) \| p_\theta(z)\right)}_{\text{"Make approximate posterior distribution close to prior"}}$$

- Training: Maximize lower bound

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Slide adapted from Feifei Li

Topics of This Lecture

- Recap: GANs
- Autoencoders
  - Motivation
  - Regularized Autoencoder
  - Denoising Autoencoder
- Variational Autoencoders (VAE)
  - Autoencoders as Generative Models
  - Intractability
  - Variational Approximation
  - Evidence Lower Bound (ELBO)
- Application Examples

Application Examples

32x32 CIFAR-10

Labeled Faces in the Wild

References

- Variational Auto-Encoders
  - D. Kingma, M. Welling, Auto-Encoding Variational Bayes, ICLR 2014.