Machine Learning for Graphs and Sequential Data

Deep Generative Models - Variational Autoencoders

Lecturer: Prof. Dr. Stephan Günnemann

www.daml.in.tum.de

Summary Term 2020

Data Analytics and
Machine Learning

Roadmap

- Chapter: Deep Generative Models
  1. Introduction
  2. Normalizing Flows
  3. **Variational Inference**
     - Latent variable models
     - Maximization using lower bounds
     - Optimizing the ELBO
     - **Variational Autoencoders**
  4. Generative Adversarial Networks
  5. Summary

$x \in \mathbb{R}^D$ $\quad \boxed{g_\lambda} \quad$ $z \in \mathbb{R}^L$ $\quad \boxed{f_\theta} \quad$ $\tilde{x} \in \mathbb{R}^D$

Recap: Latent Variable Models

- We define a generative model with latent variables
$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) d\boldsymbol{z}$$

- This latent structure allows us to define a complex distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, even though $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ are "relatively simple"

- Since the log-likelihood is intractable, we maximize the ELBO instead
$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z})\right] =: \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$$

- How do we actually define and learn such models in practice?

Designing an LVM

- In variational inference, our optimization problem is

$$\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z})\right]$$

- To define a latent variable model, we need to answer several questions
  1. What are our latent variables $\boldsymbol{z}$?
  2. What is the data $\boldsymbol{x}$?
  3. What is the prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ on the latent variables?
  4. What is the conditional likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ of the data given the latent variables?
  5. What is the variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z})$?

- These choices (especially 3, 4, 5) are to some degree arbitrary – different choices will produce different models

- We will learn about the most popular models used in practice – but many other choices are also possible!

Choosing $p_{\boldsymbol{\theta}}(\boldsymbol{z})$

- We usually choose $\boldsymbol{z}$ to be continuous
$$\boldsymbol{z} \in \mathbb{R}^L$$
- The main advantage of making $\boldsymbol{z}$ continuous is that it's easier to sample with reparametrization from continuous distributions (i.e. $q(\boldsymbol{z})$)

- We pick the simplest possible prior on $\boldsymbol{z}$ – standard normal distribution
$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$$
  - Note that we don't write $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ anymore since there are no learnable parameters $\boldsymbol{\theta}$

- We will introduce complexity to our model when designing $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$

- Picking a simple prior will significantly simplify some calculations later

Representing the Data

$p_\theta(x|z)$

- The data $\boldsymbol{x}$ depends on our application, e.g. color images are often represented as real-valued vectors

$$\boldsymbol{x} \in \mathbb{R}^D$$

- Black-and-white images can be represented as binary vectors

$$\boldsymbol{x} \in \{0,1\}^D$$

- Most examples in this week's lecture (and online) deal with images because
  - It's a popular topic – well-studied, we know what works well, code available
  - We can show pretty pictures of the results

- However, we can apply these methods across many domains – music, text, graphs, time series, data from the Large Hadron Collider, …

Choosing $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$

$\mu(z)$

- The choice of $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ depends on what data $\boldsymbol{x}$ we are modeling
- For every $\boldsymbol{z} \in \mathbb{R}^L$, we need to obtain a probability distribution over $\boldsymbol{x}$

$\psi = \{W, b\}$

- Idea: Pick a parametric distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ whose parameters are produced by some function $f_{\boldsymbol{\psi}}$ that takes $\boldsymbol{z}$ as input

$\in \mathbb{R}^2$

$x \in \mathbb{R}^2$

$z \in \mathbb{R}$

- For example, for $\boldsymbol{x} \in \mathbb{R}^D$ we could choose

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu} = f_{\boldsymbol{\psi}}(\boldsymbol{z}), \boldsymbol{I}\right)$$

$f_\psi(z) = W \cdot z + b$

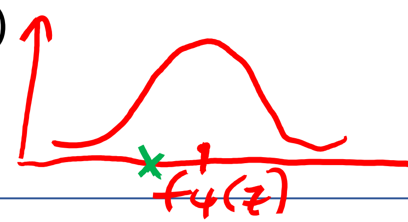
$= \mu$

$\in \mathbb{R}^{2\times 1}$

where $f_{\boldsymbol{\psi}} : \mathbb{R}^L \to \mathbb{R}^D$ is some nonlinear function (a neural network)

and $\boldsymbol{\theta} = \boldsymbol{\mu} \in \mathbb{R}^D$ are the parameters of $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$

$= f_\psi(z)$



Choosing $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$

$n_e(x)$

- Choice of $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ involves a trade-off between expressiveness and efficiency

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{x} \middle| \boldsymbol{\mu} = f_{\boldsymbol{\psi}}(\boldsymbol{z}), \boldsymbol{I}\right) = \prod_{j=1}^{D} \mathcal{N}\left(x_j \middle| \mu = f_{\boldsymbol{\psi}}(\boldsymbol{z})_j, 1\right)$$

  - Each pixel x_j is conditionally independent of the others given $\boldsymbol{z}$ (but they become dependent if we marginalize out $\boldsymbol{z}$)
  - We could have a more expressive $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ (e.g. full covariance, normalizing flow) but that would make the evaluation less efficient

$n_e(x|z) =$
$\mathcal{N}(x|\mu = f_\psi(z),$
$\Sigma = \mathrm{DIAG}(f_\psi'(z)))$

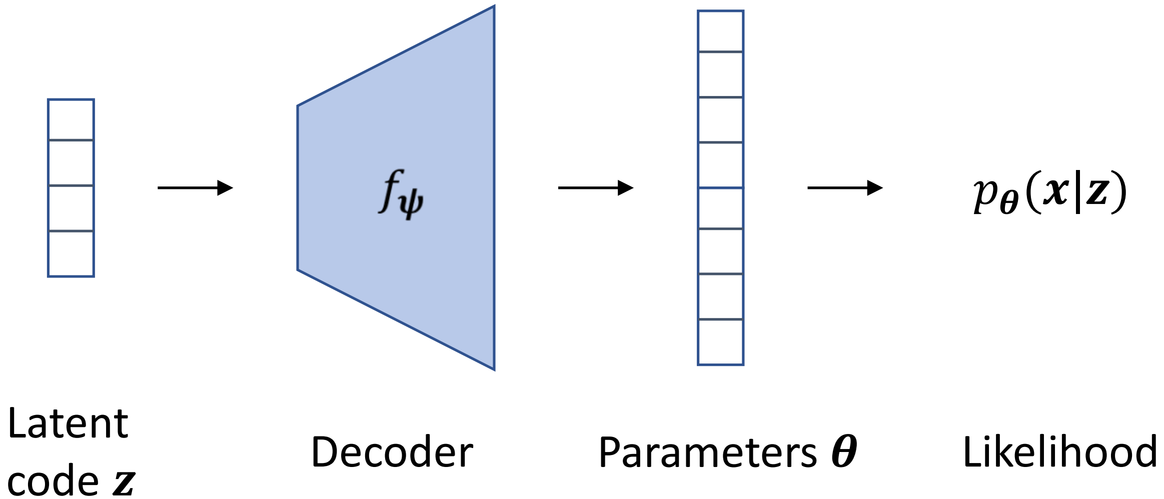
- Different data types require different likelihoods.
  - E.g., for a binary $\boldsymbol{x} \in \{0,1\}^D$ we could use

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \prod_{j=1}^{D} \text{Bernoulli}\left(x_j \middle| \underbrace{\sigma\left(f_{\boldsymbol{\psi}}(\boldsymbol{z})_j\right)}_{\theta}\right)$$

$g_\omega(y) = x$
$\omega = f_\psi(z)$
$n_1(y)$ $\mathbb{R}^D$

The Decoder f_ψ

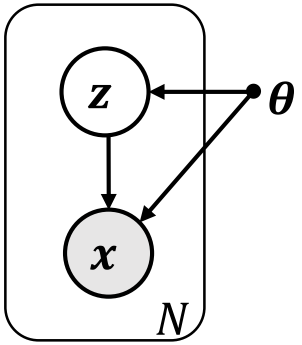
- The neural network f_ψ is often called "the decoder", since it converts the latent variable $\boldsymbol{z}$ (a.k.a. the latent code) into the parameters $\boldsymbol{\theta}$ of $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$



- We use different decoder architectures for different data types
  – E.g., a popular choice of f_ψ for images is [transposed convolution]

Choosing $q_{\boldsymbol{\phi}}(\boldsymbol{z})$

- We have specified $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and $p(\boldsymbol{z})$, the last missing component is the variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z})$



- Our dataset consists of N (i.i.d.) samples $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$

- Each sample $\boldsymbol{x}^{(i)}$ corresponds to a separate latent variable $\boldsymbol{z}^{(i)}$
  - Our variational distribution is over all $\boldsymbol{z}^{(i)}$'s: $q_{\boldsymbol{\phi}}\left(\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(N)}\right)$

- For simplicity, we use the mean field assumption

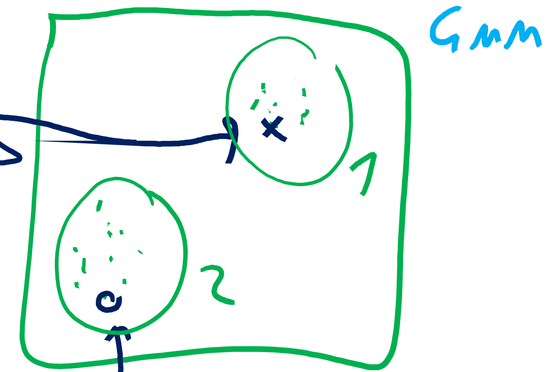
$$q_{\boldsymbol{\phi}}\left(\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(N)}\right) = \prod_{i=1}^{N} q_{\boldsymbol{\phi}^{(i)}}\left(\boldsymbol{z}^{(i)}\right)$$

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

Choosing $q_{\boldsymbol{\phi}}(z)$

$\mathrm{CAT}(0.9, 0.1) = q_{\phi^{(1)}}(z^{(1)})$

$z \sim \mathrm{CAT}(\pi)$



$q_{\phi^{(2)}}(z^{(2)})$

$= \mathrm{CAT}(0.1, 0.9)$

- Mean field assumption

$$q_{\boldsymbol{\phi}}\left(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}\right) = \prod_{i=1}^{N} q_{\boldsymbol{\phi}^{(i)}}\left(\mathbf{z}^{(i)}\right)$$

- How should we model each $q_{\boldsymbol{\phi}^{(i)}}\left(\mathbf{z}^{(i)}\right)$?

- Simple choice – multivariate normal

$$q_{\boldsymbol{\phi}^{(i)}}\left(\mathbf{z}^{(i)}\right) = \mathcal{N}\left(\mathbf{z}^{(i)} \middle| \boldsymbol{\mu}^{(i)}, \boldsymbol{\Sigma}^{(i)}\right) \quad \text{where} \;\; \boldsymbol{\phi}^{(i)} = \left\{\boldsymbol{\mu}^{(i)}, \boldsymbol{\Sigma}^{(i)}\right\}$$

  - Usually $\boldsymbol{\Sigma}^{(i)}$ is diagonal

- Another popular option – normalizing flows with forward parametrization

$p(z) = N(z|0,1) \qquad z \sim N(0,1)$

Why is Normal $q_{\phi}(z)$ a Good Choice?

$$\mathbb{E}_{z \sim q_\phi(z)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z)}\right]$$

- ELBO for a single instance $\boldsymbol{x}$

$$\boxed{\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})} = \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \log p(\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z})]$$
$$= \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z})||p(\boldsymbol{z})\right)$$

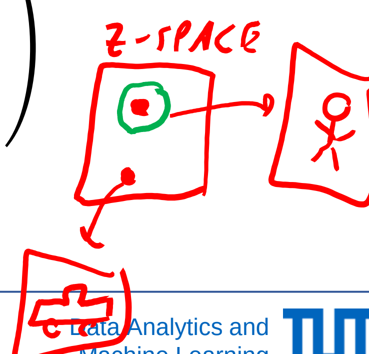
- Recall that $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- We can compute the KL divergence between two multivariate normal distributions in closed form

$$\mathbb{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z})||p(\boldsymbol{z})\right) = \frac{1}{2}\left(\mathrm{tr}(\boldsymbol{\Sigma}) + \boldsymbol{\mu}^T\boldsymbol{\mu} - \log(\det(\boldsymbol{\Sigma})) - L\right)$$

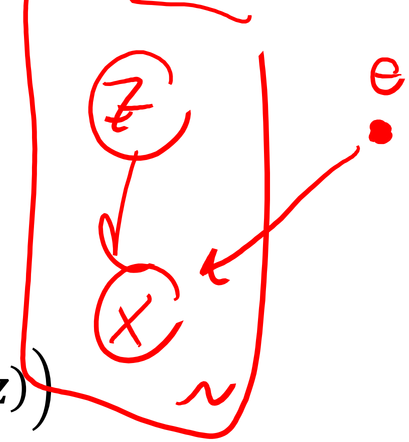
- Even simpler for diagonal covariance $\boldsymbol{\Sigma} = \mathrm{diag}\left(\boldsymbol{\sigma}^2\right)$ (where $\boldsymbol{\sigma}^2 \in \mathbb{R}_+^L$)

$$\mathbb{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z})||p(\boldsymbol{z})\right) = \frac{1}{2}\left(\sum_{j=1}^{L}(\sigma_j^2 + \mu_j^2 - \log \sigma_j^2 - 1)\right)$$

$$\log p_\theta(x) = \boxed{\mathrm{ELBO}} + \mathrm{KL}\left(q_\phi(z) \,||\, p(z|x)\right)$$



ELBO for Multiple Samples



- ELBO for a single instance $\boldsymbol{x}$

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}) || p(\boldsymbol{z})\right)$$

- ELBO for the entire dataset $\left\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}\right\}$

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{N} \sum_{i}^{N} \mathcal{L}_i\left(\boldsymbol{\theta}, \boldsymbol{\phi}^{(i)}\right)$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left[\mathbb{E}_{\boldsymbol{z}^{(i)} \sim q_{\boldsymbol{\phi}^{(i)}}\left(\boldsymbol{z}^{(i)}\right)}\left[\log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)} \middle| \boldsymbol{z}^{(i)}\right)\right] - \mathbb{KL}\left(q_{\boldsymbol{\phi}^{(i)}}\left(\boldsymbol{z}^{(i)}\right) || p(\boldsymbol{z})\right)\right]$$

- We have to learn a separate parameter $\boldsymbol{\phi}^{(i)}$ for each instance i
  - If the number of samples N is large, this is extremely expensive
- What if we are interested in the latent features of a new sample $\boldsymbol{x}^{\text{new}}$?
  - We will need to learn $\boldsymbol{\phi}^{\text{new}}$ from scratch

- Can we do better than this?

Amortized Inference

- We need to find the optimal parameter $\boldsymbol{\phi}^{(i)}_{\text{optimal}}$ for every sample $\boldsymbol{x}^{(i)}$

- Standard approach: Solve the optimization problem for each $i = 1, \dots, N$

$$\boldsymbol{\phi}^{(i)}_{\text{optimal}} = \underset{\boldsymbol{\phi}^{(i)}}{\operatorname{argmax}}\, \mathcal{L}_i\left(\boldsymbol{\theta}, \boldsymbol{\phi}^{(i)}\right)$$

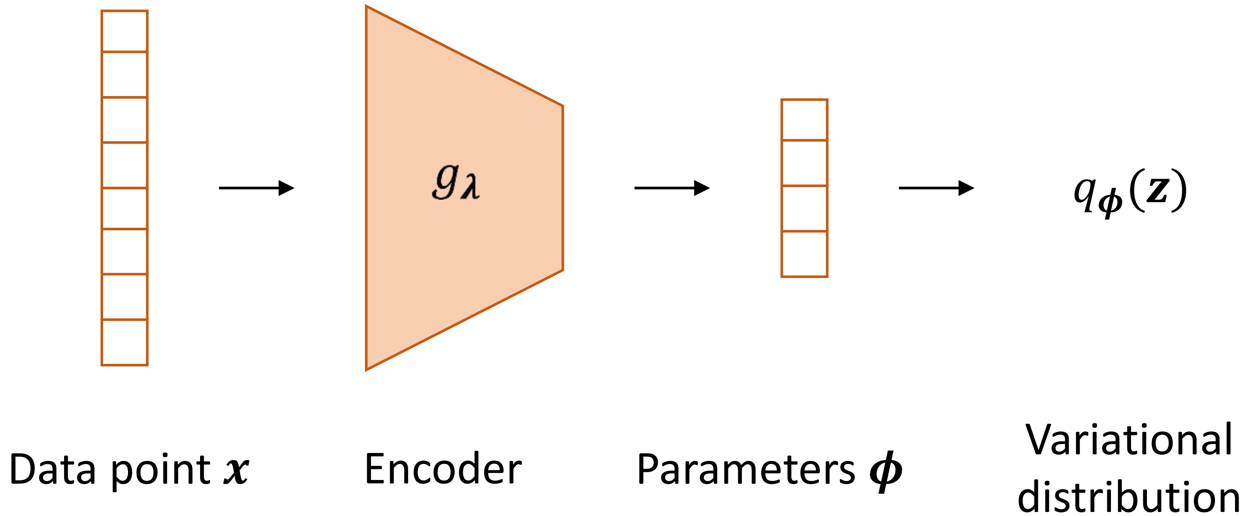
- Better idea: Train a neural network $g_{\boldsymbol{\lambda}}$ that tries to map every $\boldsymbol{x}^{(i)}$ in the training set to its optimal parameters $\boldsymbol{\phi}^{(i)}_{\text{optimal}}$

$$\max_{\boldsymbol{\lambda}} \frac{1}{N} \sum_{i}^{N} \mathcal{L}_i\left(\boldsymbol{\theta}, \underbrace{g_{\boldsymbol{\lambda}}\left(\boldsymbol{x}^{(i)}\right)}_{\approx \phi^{(i)}_{\text{optimal}}}\right)$$

- We use the same g_{λ} for every sample $\boldsymbol{x}^{(i)}$, and can even use for new samples that we haven't seen during training

The Encoder g_λ

- We call the NN g_λ "the encoder", since it converts a data point $\boldsymbol{x}$ into the parameters $\boldsymbol{\phi}$ that define the distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ over the latent code $\boldsymbol{z}$



- We use different encoder architectures for different data types
  - E.g., a popular choice of g_λ for images are convolutional NNs

Putting Everything Together

- ELBO for a single sample

$$\mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\lambda}) := \boxed{\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]} - \mathbb{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z})||p(\boldsymbol{z})\right)$$
$$\text{where } \boldsymbol{\theta} = f_{\boldsymbol{\psi}}(\boldsymbol{z}) \text{ and } \boldsymbol{\phi} = g_{\boldsymbol{\lambda}}(\boldsymbol{x})$$

- Recipe for optimizing the ELBO
  1. Compute $\boldsymbol{\phi} = g_{\boldsymbol{\lambda}}(\boldsymbol{x})$
  2. Compute an MC estimate of ELBO; often done using a single sample, i.e.
     a) Draw $\boldsymbol{z}' \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})$ with reparametrization
     b) Compute $\boldsymbol{\theta} = f_{\boldsymbol{\psi}}(\boldsymbol{z}')$
     c) ELBO: $\mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\lambda}) \approx \boxed{\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}')} - \mathbb{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z})||p(\boldsymbol{z})\right)$
  3. Backpropagate (compute $\nabla_{\boldsymbol{\psi}}\mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\lambda})$ and $\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\lambda})$)
  4. Update the NN weights $\boldsymbol{\psi}$ and $\boldsymbol{\lambda}$ using gradient ascent

Recall: Reparametrization Trick

- Our expectation depends on the parameters that we are optimizing

$$\mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\lambda}) := \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - \mathbb{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}) || p(\mathbf{z})\right)$$

$$\text{where } \boldsymbol{\theta} = f_{\boldsymbol{\psi}}(\mathbf{z}) \text{ and } \boldsymbol{\phi} = g_{\boldsymbol{\lambda}}(\mathbf{x})$$

- This means that we need to sample from $q_{\boldsymbol{\phi}}(\mathbf{z})$ with reparametrization

1. $\boldsymbol{\epsilon} \sim b(\boldsymbol{\epsilon})$
2. $\mathbf{z}' = T(\boldsymbol{\epsilon}, \boldsymbol{\phi})$

- E.g., for $q_{\boldsymbol{\phi}}(\mathbf{z})$ multivariate normal with diagonal covariance (Slide 98)

$$q_{\boldsymbol{\phi}}(\mathbf{z}) = \mathcal{N}\left(\mathbf{z} \middle| \boldsymbol{\mu}, \text{diag}\left(\boldsymbol{\sigma}^2\right)\right)$$

1. $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{I})$
2. $\mathbf{z}' = \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}$

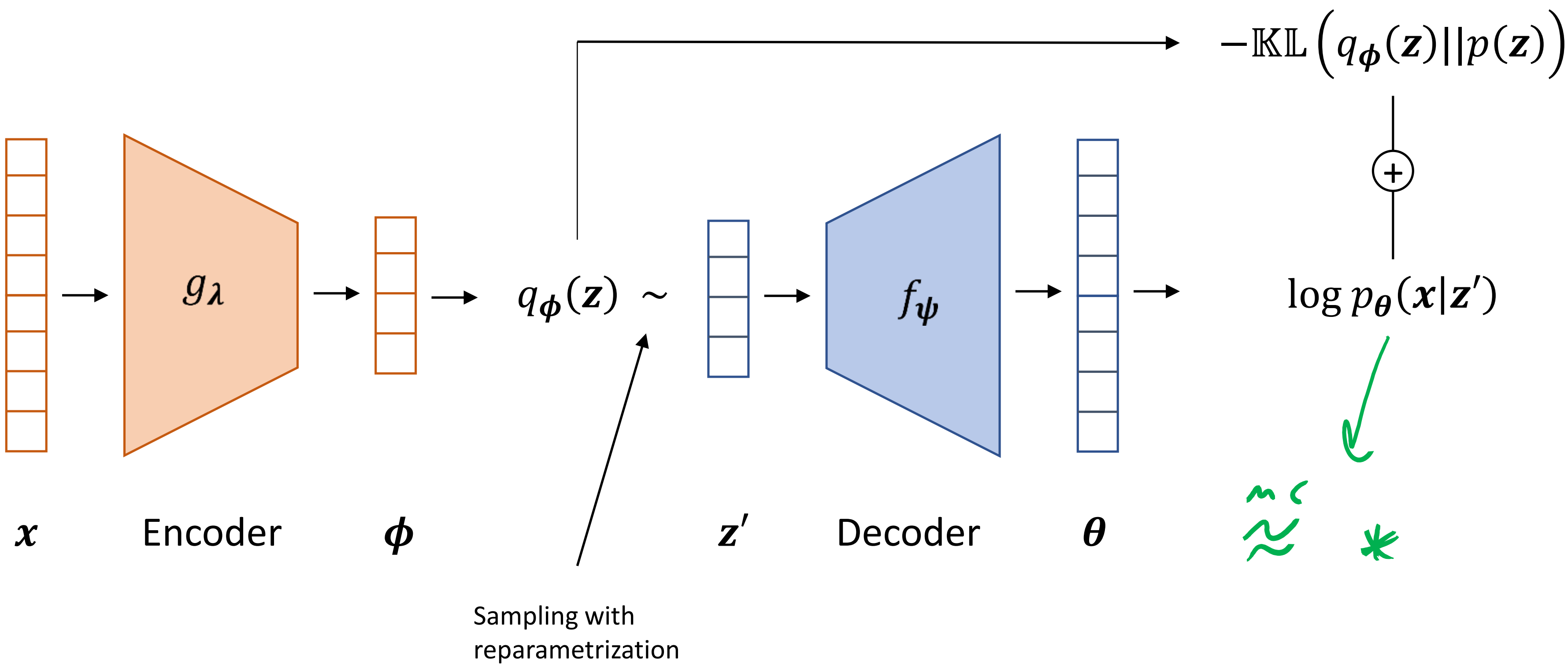
$\phi = g_\lambda(x)$

Variational Autoencoder (VAE)

*

$$\mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\lambda}) := \boxed{\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]} - \mathbb{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}) || p(\boldsymbol{z})\right)$$

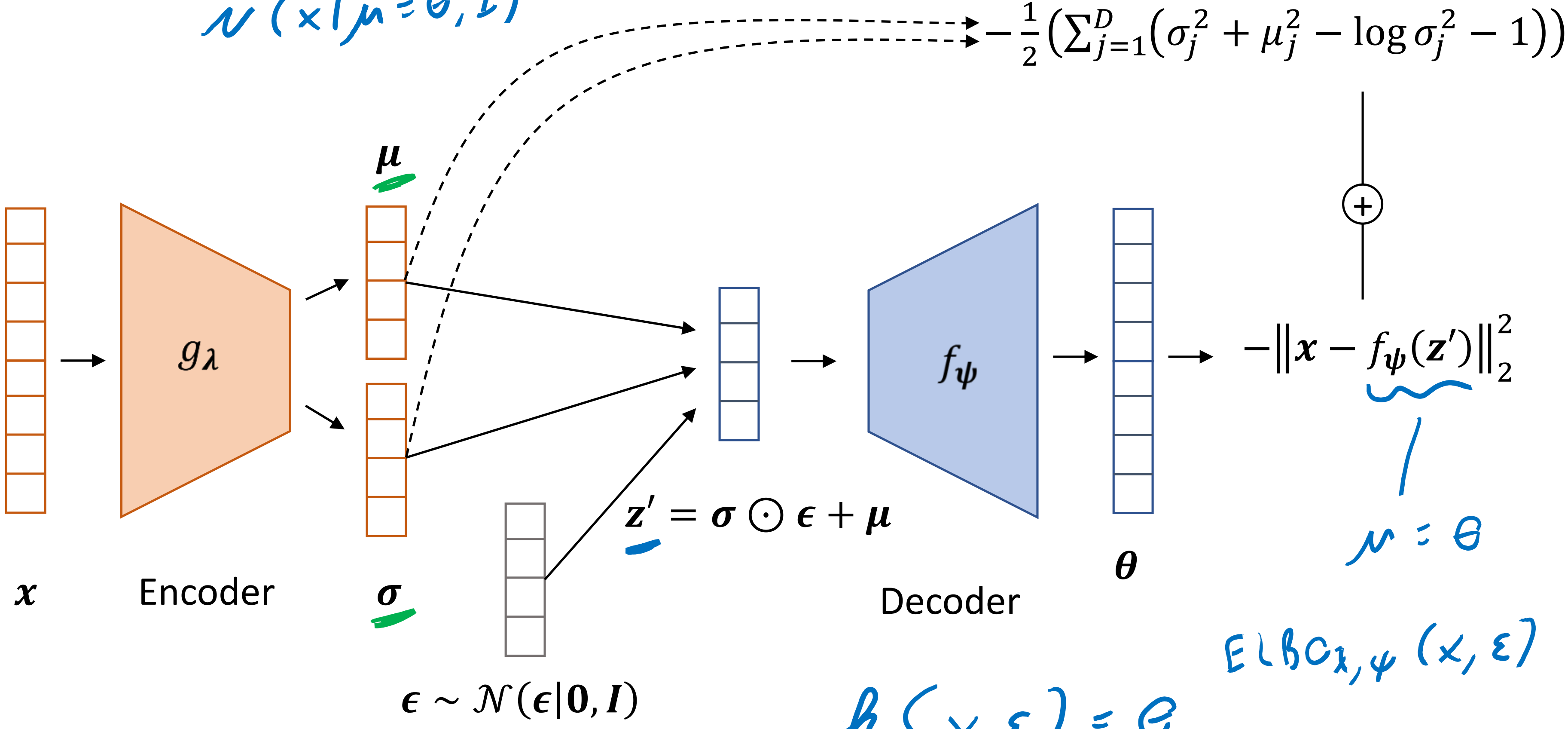
$$\text{where } \quad \boldsymbol{\phi} = g_{\boldsymbol{\lambda}}(\boldsymbol{x}) \;\text{ and }\; \boldsymbol{\theta} = f_{\boldsymbol{\psi}}(\boldsymbol{z})$$



VAE with Gaussian $p_\theta(x|z)$ and $q_\phi(z)$

- Using our choices of $p_\theta(x|z)$ and $q_\phi(z)$ from Slides 94 & 98

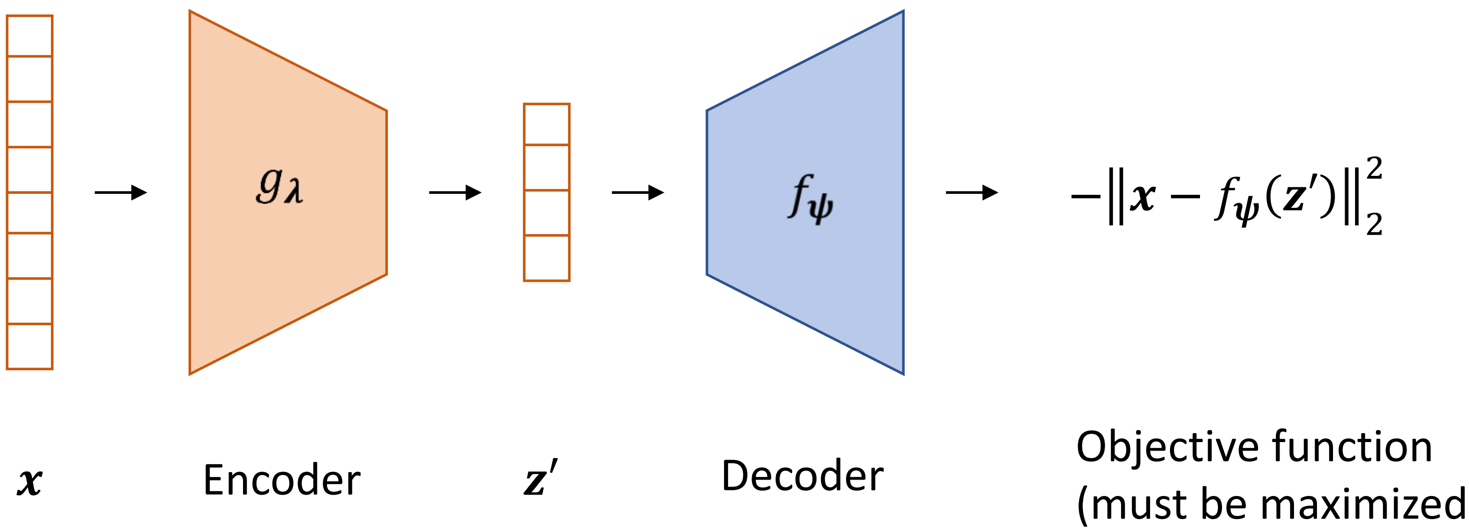
$\mathcal{N}(x|\mu=\theta, I)$ $\quad z' \sim \mathcal{N}(z|\mu,\sigma)$



Standard Autoencoders

$$\ell(\boldsymbol{\psi}, \boldsymbol{\lambda}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}')$$
$$\text{where} \quad \boldsymbol{z}' = g_{\boldsymbol{\lambda}}(\boldsymbol{x}) \ \text{ and } \boldsymbol{\theta} = f_{\boldsymbol{\psi}}(\boldsymbol{z}')$$

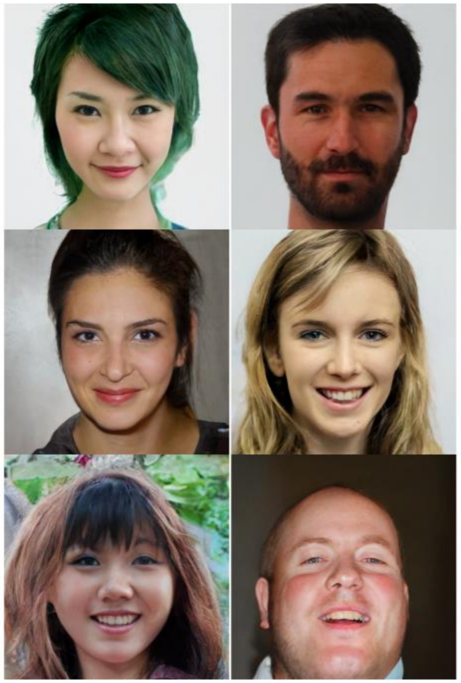
- When $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}')$ is Gaussian distribution, we get



- Standard AE are not generative models – they can only reconstruct the data
- Standard AE learns a single $\boldsymbol{z}'$ for each $\boldsymbol{x}$, while VAE learns a distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z})$

Generating Data with a VAE

- Remember: Once we have trained our generative model (i.e. we know the parameters $\boldsymbol{\psi}$ of our decoder) we can use it to generate new data
  1. Sample $\boldsymbol{z}' \sim p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$
  2. Sample $\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}') = \mathcal{N}\left(\boldsymbol{x}\middle|f_{\boldsymbol{\psi}}(\boldsymbol{z}'), \boldsymbol{I}\right)$
- Note that we don't need the encoder when sampling new data



Source: Razavi et al., 2019
Generating Diverse High-Fidelity Images with VQ-VAE-2

Questions – VAE

1. Assume that each data point is represented by a vector $\boldsymbol{x} \in \{1, 2, \dots, C\}^D$. What distribution would you pick for $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$? How can we parametrize this distribution with a neural network?

2. Assume that each datapoint $\boldsymbol{x}$ is represented by a variable-length sequence of real numbers $\boldsymbol{x}_i \in \mathbb{R}^{D_i}$ (D_i might be different for different i's). What NN architecture could we use for the encoder $g_{\boldsymbol{\lambda}}$ in this case?

3. Slide 94: Assume that we choose to model $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ with a normalizing flow. Should we use forward or reverse parametrization? Why?

4. Slide 97: Assume that we choose to model $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ with a normalizing flow. Should we use forward or reverse parametrization? Why?

5. Slide 106: How can we ensure that the vector $\boldsymbol{\sigma}$ produced by the encoder $g_{\boldsymbol{\lambda}}$ is always positive?

Reading Materials

- Sections 1.2 – 1.7 and 2.1 – 2.6 of the PhD thesis of Diedrik P. Kingma cover essentially the same content as our lecture
  - https://pure.uva.nl/ws/files/17891313/Thesis.pdf