

# Variational Autoencoders

Machine Learning and Data Mining, 2020

Artem Maevskiy

National Research University Higher School of Economics



LAMBDA • HSE

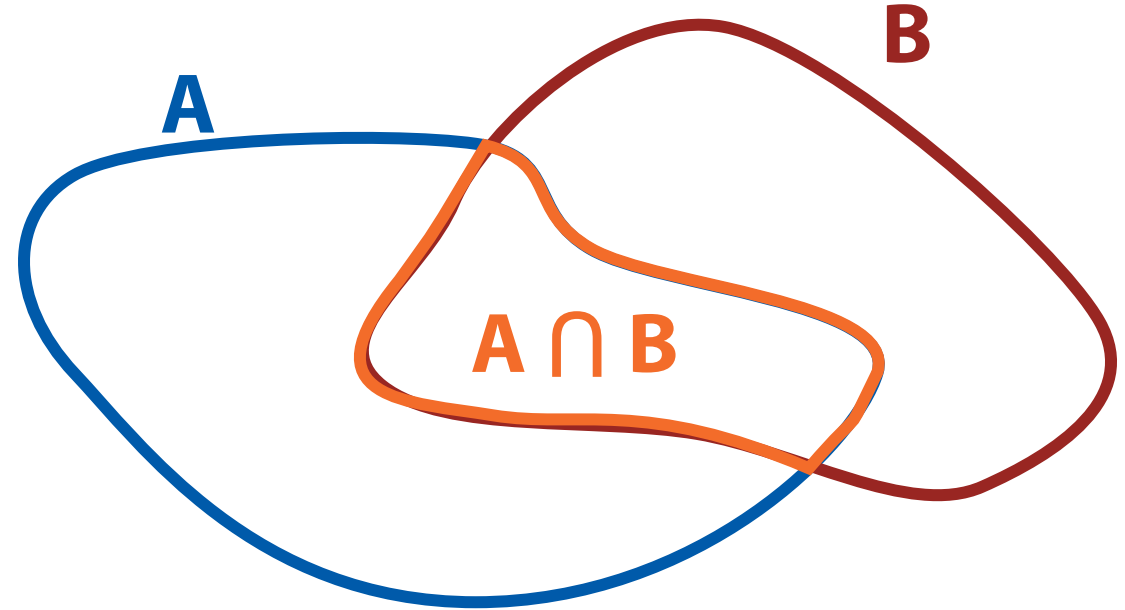
December 04, 2020

# A few definitions



# Conditional probability (recap)

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$



For PDF:  $p(x|y) = \frac{p(x, y)}{p(y)}$

– i.e. we're renormalizing  $p(x, y)$  as a distribution of only  $x$  for some fixed  $y$

# Entropy

- ▶ Entropy of a distribution:

$$\mathcal{H}(p) = -\mathbb{E}_{x \sim p(x)} \log p(x)$$

- measure uncertainty of the random variable  $x$
- e.g., for the normal distribution it is proportional to  $\log \sigma^2$  (the wider the distribution, the more uncertainty there is)
- ▶ Note that it equals to the expected negative log likelihood **of the correct hypothesis** (i.e., when the data is actually distributed as  $p$ )

# Cross-entropy

- ▶ Cross-entropy between two distributions:

$$\mathcal{H}(p, q) = -\mathbb{E}_{x \sim p(x)} \log q(x)$$

- ▶ May be thought of as the expected negative log likelihood of  $q$  when the data is actually distributed as  $p$

# Kullback–Leibler divergence

$$D_{\text{KL}}(p \parallel q) \equiv \mathcal{H}(p, q) - \mathcal{H}(p)$$

$$= \mathbb{E}_{x \sim p(x)} \log \frac{p(x)}{q(x)}$$

- Interpretation: expected logarithm of the **likelihood ratio** between the true data distribution and distribution  $q$  when the data is actually distributed as  $p$

$$D_{\text{KL}}(p \parallel q) \geq 0$$

$$D_{\text{KL}}(p \parallel q) = 0 \iff p = q \text{ (almost everywhere)}$$

# Generative modelling

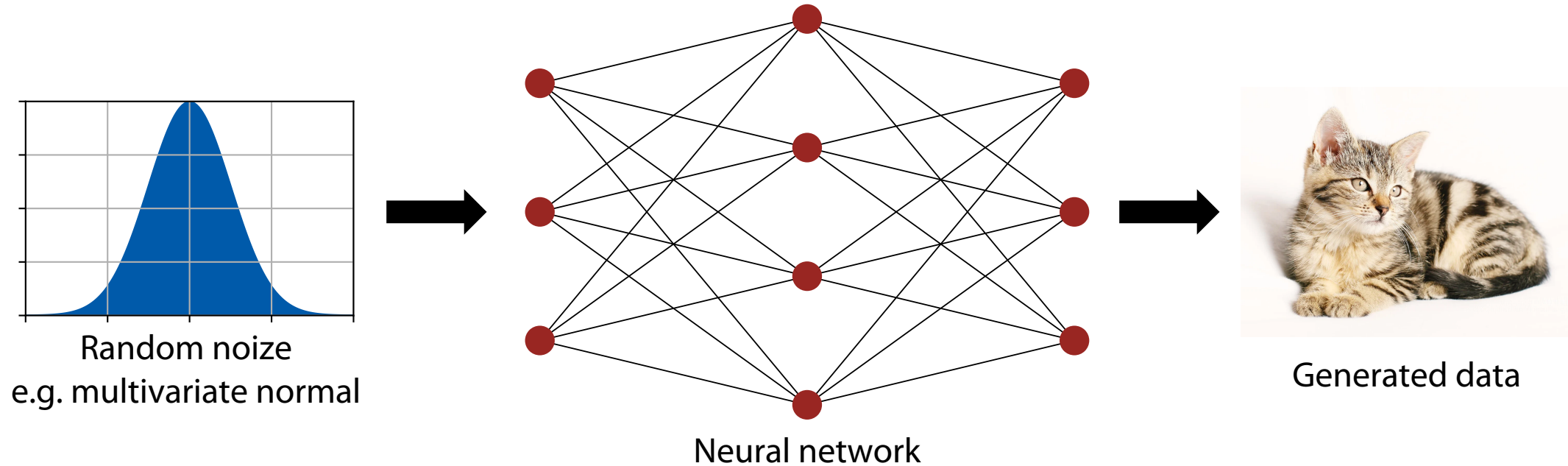


# Problem setup (recap)

- ▶ Given a set of training objects  $\{x_i\}$
- ▶ We want to approximate their population distribution  $p_{\text{data}}(x)$
- ▶ E.g. with some distribution  $p_{\theta}(x)$  parametrized with  $\theta$
- ▶ To be able to sample new objects  $x' \sim p_{\theta}$ , that are similar to  $\{x_i\}$



# How can a neural network generate data? (recap)



- This makes the generated object being a **differentiable function** of the network parameters

# Modelling the probability density

- ▶ Generated object:  $x' = G_{\theta}(z)$ ,
- ▶ Latent code:  $z \sim p_z$  (sampled from some fixed distribution)

# Modelling the probability density

- ▶ Generated object:  $x' = G_{\theta}(z)$ ,
- ▶ Latent code:  $z \sim p_z$  (sampled from some fixed distribution)
- ▶ We can treat our model as a model for the probability density, e.g.:

$$p_{\theta}(x | z) = \mathcal{N}(x | \mu = G_{\theta}(z), \Sigma = \mathbb{I}\sigma^2)$$

I.e. the network generates not just a single object,  
but rather the average object for the given latent code  $z$

fixed parameter

# Modelling the probability density

- ▶ Generated object:  $x' = G_\theta(z)$ ,
- ▶ Latent code:  $z \sim p_z$  (sampled from some fixed distribution)
- ▶ We can treat our model as a model for the probability density, e.g.:

$$p_\theta(x | z) = \mathcal{N}(x | \mu = G_\theta(z), \Sigma = \mathbb{I}\sigma^2)$$

I.e. the network generates not just a single object,  
but rather the average object for the given latent code  $z$

fixed parameter

$$p_\theta(x) = \int p_\theta(x, z) dz = \int p_\theta(x | z) p_z(z) dz = \mathbb{E}_{z \sim p_z} p_\theta(x | z)$$

prior on  $z$

# How can we train it?

- ▶ To train the model, we'd want to maximize the expected log likelihood:

$$L = \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\theta}(x) \rightarrow \max_{\theta}$$

# How can we train it?

- ▶ To train the model, we'd want to maximize the expected log likelihood:

$$L = \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\theta}(x) \rightarrow \max_{\theta}$$

- ▶ But  $p_{\theta}(x) = \mathbb{E}_{z \sim p_z} p_{\theta}(x | z)$ , i.e. it is not expressed in a closed form
  - we only have closed forms for  $p_{\theta}(x | z)$  and  $p_z(z)$

# How can we train it?

- ▶ To train the model, we'd want to maximize the expected log likelihood:

$$L = \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\theta}(x) \rightarrow \max_{\theta}$$

- ▶ But  $p_{\theta}(x) = \mathbb{E}_{z \sim p_z} p_{\theta}(x | z)$ , i.e. it is not expressed in a closed form
  - we only have closed forms for  $p_{\theta}(x | z)$  and  $p_z(z)$
- ▶ Sampling  $z \sim p_z, x \sim p_{\text{data}}$  and then maximizing the likelihood is typically not very productive
  - Like this,  $z$  contains no information about  $x$ , and the network will likely learn just to ignore it and always predict the same object

# The posterior

- ▶ Assume we're able to calculate (and sample from) the posterior  $p_\theta(z|x)$
- ▶ Note that:

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



# The posterior

- ▶ Assume we're able to calculate (and sample from) the posterior  $p_\theta(z|x)$
- ▶ Note that:

$$\begin{aligned}\log p_\theta(x) &= \mathbb{E}_{z \sim p_\theta(z|x)} \log \left[ p_\theta(x) \frac{p_\theta(z|x)}{p_\theta(z|x)} \right] \\ &= \mathbb{E}_{z \sim p_\theta(z|x)} [\log p_\theta(x, z) - \log p_\theta(z|x)]\end{aligned}$$

# The posterior

- ▶ Assume we're able to calculate (and sample from) the posterior  $p_\theta(z|x)$
- ▶ Note that:

$$\begin{aligned}\log p_\theta(x) &= \mathbb{E}_{z \sim p_\theta(z|x)} \log \left[ p_\theta(x) \frac{p_\theta(z|x)}{p_\theta(z|x)} \right] \\ &= \mathbb{E}_{z \sim p_\theta(z|x)} [\log p_\theta(x, z) - \log p_\theta(z|x)] \\ &= \mathbb{E}_{z \sim p_\theta(z|x)} \log p_\theta(x, z) + \mathcal{H}(p_\theta(z|x))\end{aligned}$$

# The posterior

- ▶ Assume we're able to calculate (and sample from) the posterior  $p_\theta(z|x)$
- ▶ Note that:

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

$$= \mathbb{E}_{z \sim p_\theta(z|x)} [\log p_\theta(x, z) - \log p_\theta(z|x)]$$

$$= \underbrace{\mathbb{E}_{z \sim p_\theta(z|x)} \log p_\theta(x, z)}_{\text{red underline}} + \underbrace{\mathcal{H}(p_\theta(z|x))}_{\text{blue underline}}$$

So, for the log-likelihood we're sampling not all  $z$  values,  
but only those corresponding to this particular  $x$

Maximizing this encourages placing high  
probability mass on many  $z$  values that  
could've generated  $x$

# Approximate inference



# Approximate inference

- ▶ In practice,  $p_{\theta}(z|x)$  is typically intractable
- ▶ Let's try to approximate it with another parametric distribution  $q_{\phi}(z|x)$ 
  - E.g.,  $q_{\phi}(z|x) = \mathcal{N}(z|\mu_{\phi}(x), \mathbb{I}\sigma_{\phi}^2(x))$ , where  $\mu_{\phi}$  and  $\sigma_{\phi}^2$  are outputs of a neural network
- ▶ And use it for the likelihood calculation, i.e.:

$$[\log p_{\theta}(x)]_{\text{approx},\phi} = \mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x, z) + \mathcal{H}(q_{\phi}(z|x))$$

# Approximate inference

- ▶ Let's check how bad this approximation is:

$$\begin{aligned} \log p_{\theta}(x) - [\log p_{\theta}(x)]_{\text{approx}, \phi} &= \\ &= \log p_{\theta}(x) - \mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x, z) - \mathcal{H} \left( q_{\phi}(z|x) \right) \end{aligned}$$

# Approximate inference

- ▶ Let's check how bad this approximation is:

$$\begin{aligned} \log p_{\theta}(x) - [\log p_{\theta}(x)]_{\text{approx}, \phi} &= \\ &= \log p_{\theta}(x) - \mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x, z) - \mathcal{H}(q_{\phi}(z|x)) \\ &= \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x) - \log p_{\theta}(x, z) + \log q_{\phi}(z|x)] \end{aligned}$$

# Approximate inference

- ▶ Let's check how bad this approximation is:

$$\begin{aligned} & \log p_{\theta}(x) - [\log p_{\theta}(x)]_{\text{approx}, \phi} = \\ &= \log p_{\theta}(x) - \mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x, z) - \mathcal{H}(q_{\phi}(z|x)) \\ &= \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x) - \log p_{\theta}(x, z) + \log q_{\phi}(z|x)] \\ &= \mathbb{E}_{z \sim q_{\phi}(z|x)} [-\log p_{\theta}(z|x) + \log q_{\phi}(z|x)] \\ &= D_{\text{KL}}(q_{\phi}(z|x) \| p_{\theta}(z|x)) \geq 0 \end{aligned}$$



# Approximate inference

- ▶ We've shown that:

$$\log p_{\theta}(x) - [\log p_{\theta}(x)]_{\text{approx}, \phi} = D_{\text{KL}}(q_{\phi}(z|x) \| p_{\theta}(z|x)) \geq 0$$

- ▶ I.e., our approximate log-likelihood is the **lower bound** for the true log-likelihood
  - Also called evidence lower bound (ELBO) or variational lower bound

# Approximate inference

- ▶ We've shown that:

$$\log p_{\theta}(x) - [\log p_{\theta}(x)]_{\text{approx}, \phi} = D_{\text{KL}}(q_{\phi}(z|x) \| p_{\theta}(z|x)) \geq 0$$

- ▶ I.e., our approximate log-likelihood is the **lower bound** for the true log-likelihood
  - Also called evidence lower bound (ELBO) or variational lower bound
- ▶ The better  $q$  approximates the posterior – the closer the bound is to the actual log-likelihood
- ▶ Also, if we maximize the lower bound, we'll maximize the likelihood as well!

# Alternative form

$$\text{ELBO} = [\log p_{\theta}(x)]_{\text{approx.}, \phi} = \mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x, z) + \mathcal{H}(q_{\phi}(z|x))$$

# Alternative form

$$\begin{aligned}\text{ELBO} &= [\log p_\theta(x)]_{\text{approx.}, \phi} = \mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x, z) + \mathcal{H}(q_\phi(z|x)) \\ &= \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)]\end{aligned}$$

# Alternative form

$$\begin{aligned}\text{ELBO} &= [\log p_{\theta}(x)]_{\text{approx.}, \phi} = \mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x, z) + \mathcal{H}(q_{\phi}(z|x)) \\ &= \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z) + \log p(z) - \log q_{\phi}(z|x)] \\ &= \mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x|z) - D_{KL}(q_{\phi}(z|x) \| p(z)) \rightarrow \max_{\theta, \phi}\end{aligned}$$

# Alternative form

$$\text{ELBO} = [\log p_{\theta}(x)]_{\text{approx.}, \phi} = \mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x, z) + \mathcal{H}(q_{\phi}(z|x))$$

$$= \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z) + \log p(z) - \log q_{\phi}(z|x)]$$

$$= \underbrace{\mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x|z)}_{\text{Data term}} - \underbrace{D_{KL}(q_{\phi}(z|x) \| p(z))}_{\text{Regularizer}} \rightarrow \max_{\theta, \phi}$$

**Data term**

**Regularizer**

# Variational autoencoder (VAE)



# Variational autoencoder

- ▶ Let's make use our choices for  $p_\theta(x | z)$ ,  $p_z(z)$  and  $q_\phi(z | x)$ :

$$p_z(z) = \mathcal{N}(z | 0, \mathbb{I})$$

$$p_\theta(x | z) = \mathcal{N}(x | \mu = G_\theta(z), \Sigma = \mathbb{I}\sigma^2)$$

$$q_\phi(z|x) = \mathcal{N}(z | \mu_\phi(x), \mathbb{I}\sigma_\phi^2(x))$$

- ▶ And see how we can optimize the two ELBO terms:

$$\text{ELBO} = \underbrace{\mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x|z)}_{\text{Data term}} - \underbrace{D_{KL}(q_\phi(z|x) || p(z))}_{\text{Regularizer}} \rightarrow \max_{\theta, \phi}$$



# The data term


$$\mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x|z)$$

- ▶ Take object  $x$  from the dataset
- ▶ Calculate the posterior  $q_\phi(z|x)$
- ▶ Sample latent code  $z_\phi$  from the posterior
- ▶ Calculate the log-likelihood for this pair  $(x, z_\phi)$ :

$$p_z(z) = \mathcal{N}(z | 0, \mathbb{I})$$

$$p_\theta(x | z) = \mathcal{N}(x | \mu = G_\theta(z), \Sigma = \mathbb{I}\sigma^2)$$

$$q_\phi(z|x) = \mathcal{N}(z | \mu_\phi(x), \mathbb{I}\sigma_\phi^2(x))$$

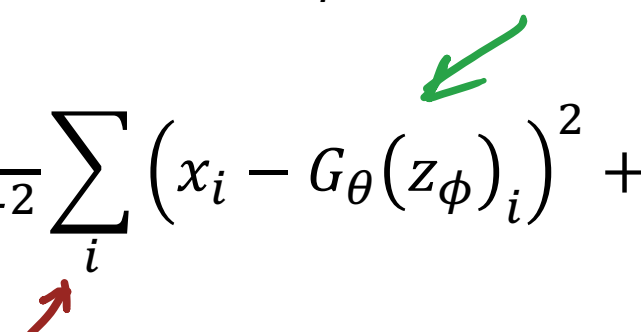
$$\log p_\theta(x|z) = -\frac{1}{2\sigma^2} \sum_i \left( x_i - G_\theta(z_\phi)_i \right)^2 + \text{const}$$


Sum over the components  
of the data vector

# The data term

$$\mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x|z)$$


- ▶ Take object  $x$  from the dataset
- ▶ Calculate the posterior  $q_\phi(z|x)$
- ▶ Sample latent code  $z_\phi$  from the posterior
- ▶ Calculate the log-likelihood for this pair  $(x, z_\phi)$ :

$$\log p_\theta(x|z) = -\frac{1}{2\sigma^2} \sum_i \left( x_i - G_\theta(z_\phi)_i \right)^2 + \text{const}$$


Sum over the components  
of the data vector

$$p_z(z) = \mathcal{N}(z | 0, \mathbb{I})$$

$$p_\theta(x | z) = \mathcal{N}(x | \mu = G_\theta(z), \Sigma = \mathbb{I}\sigma^2)$$

$$q_\phi(z|x) = \mathcal{N}(z | \underbrace{\mu_\phi(x), \mathbb{I}\sigma_\phi^2(x)}_{\text{encoder}})$$


"decoder"

"encoder"

# The data term

$$\mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x|z)$$

- ▶ Take object  $x$  from the dataset
- ▶ Calculate the posterior  $q_\phi(z|x)$
- ▶ Sample latent code  $z_\phi$  from the posterior
- ▶ Calculate the log-likelihood for this pair  $(x, z_\phi)$ :

$$p_z(z) = \mathcal{N}(z | 0, \mathbb{I})$$

$$p_\theta(x | z) = \mathcal{N}(x | \mu = G_\theta(z), \Sigma = \mathbb{I}\sigma^2)$$

$$q_\phi(z|x) = \mathcal{N}(z | \underbrace{\mu_\phi(x)}_{\text{"encoder"}}, \mathbb{I}\sigma_\phi^2(x))$$

$$\log p_\theta(x|z) = -\frac{1}{2\sigma^2} \sum_i \left( x_i - G_\theta(z_\phi)_i \right)^2 + \text{const}$$

Sum over the components  
of the data vector

How do we backpropagate  
through  $z$  (to optimize wrt  $\phi$ )?

# Backpropagating through randomness

## Reparametrization trick:

- ▶ To sample  $z_\phi \sim \mathcal{N}(\mu_\phi(x), \sigma_\phi^2(x))$ ,
- ▶ we first sample  $\xi \sim \mathcal{N}(0,1)$ ,
- ▶ then set  $z_\phi = \xi \cdot \sigma_\phi(x) + \mu_\phi(x)$

# Regularizer term

$$\text{ELBO} = \underbrace{\mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x|z)}_{\text{Data term}} - \underbrace{D_{KL}(q_\phi(z|x) \| p(z))}_{\text{Regularizer}} \rightarrow \max_{\theta, \phi}$$

- ▶  $p_z$  is  $\mathcal{N}(0, \mathbb{I})$
- ▶  $q_\phi(z|x)$  is a normal with  $\mu = \mu_\phi(x)$ ,  $\Sigma = \mathbb{I}\sigma_\phi^2(x)$
- ▶ KL divergence between them can be calculated analytically
- ▶ Prove that:
  - KL is additive for factorizing distributions  $D_{KL}(p_x p_y \| q_x q_y) = D_{KL}(p_x \| q_x) + D_{KL}(p_y \| q_y)$ ,
  - KL between two univariate normal distributions is:

$$D_{KL}(\mathcal{N}(\mu_1, \sigma_1^2) \| \mathcal{N}(\mu_2, \sigma_2^2)) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_2 - \mu_1)^2}{2\sigma_2^2} - \frac{1}{2}$$

# Discussion



# VAE vs GANs

- ▶ In the tasks of image generation GANs are typically better
  - VAEs tend to produce blurry results due to the nature of the MSE loss
  - Note that MSE loss between images does not reflect our perception of image quality or similarity:

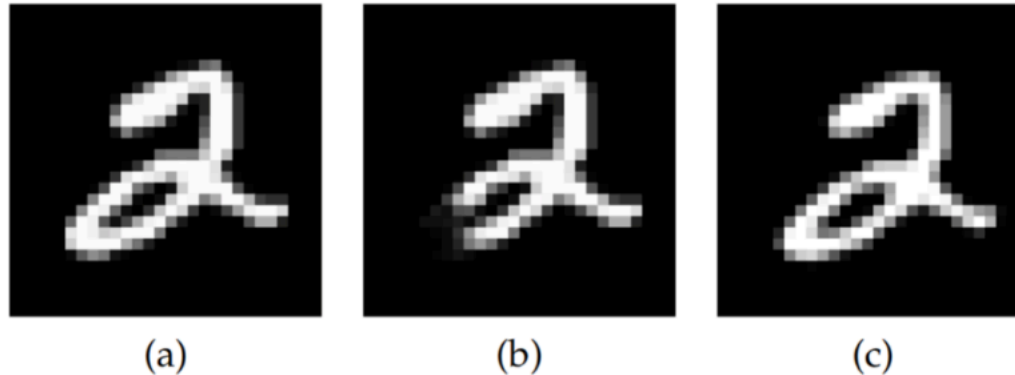


Image (b) — slightly altered image (a), image (c) — image (a) shifted by several pixels.  
Under MSE metric, image (b) is much closer to (a), than (c) to (a).

- There are some further advancements in VAEs that perform better (e.g., adversarial VAE)

# VAEs vs GANs

- ▶ VAE is easier to train – no min-max game, just a single optimization objective
- ▶ The encoder gives you the mapping from objects to the latent representation
  - This lets you do things like interpolation between objects, analyzing latent space, etc.
- ▶ VAEs give you explicit access to the estimated data PDF



# Bayesian neural networks in a nutshell

- ▶ Variational inference and ELBO optimization are very powerful techniques
- ▶ Also applied in bayessian neural networks
- ▶ The main idea is to treat the weights as random variables, with some prior distribution on them, i.e. the model is  $p(y|x) = \mathbb{E}_{w \sim p(w)}[p(y|x, w)]$
- ▶ Approximate posterior  $p(w|X, Y)$  with a parametric distribution  $q_\phi(w)$ 
  - Something easy to sample from - allowing for the reparametrisation trick, and allowing the analytic calculation of KL divergence wrt prior
- ▶ Optimize ELBO:

$$\text{ELBO} = \mathbb{E}_{w \sim q_\phi} \log p(y|x, w) - D_{\text{KL}}(q_\phi(w) \parallel p(w))$$

# Bayesian neural networks in a nutshell

- ▶ Having found the approximate posterior  $q_{\phi}(w)$ , use expected  $w$  for prediction
- ▶ May also sample different weights from  $q_{\phi}(w)$  to estimate the uncertainty of the prediction
- ▶ Different priors favor different properties of the network
  - E.g. log-uniform prior helps removing noisy weights and thus finding sparse solutions (as was shown in <https://arxiv.org/abs/1701.05369>)

# Thank you!



[amaevskij@hse.ru](mailto:amaevskij@hse.ru)



SiLiKhon



hse\_lambda

Artem Maevskiy