Tutorial

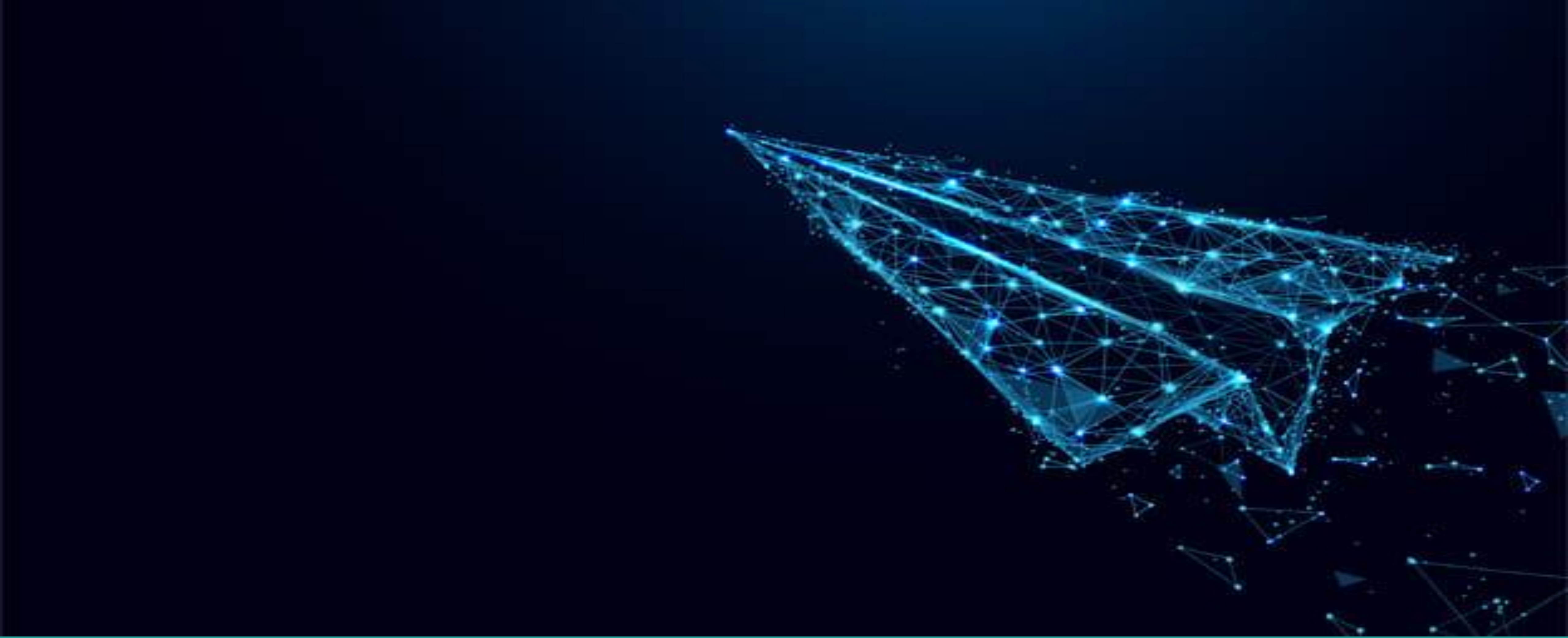
Synthetic Smart Meter Data Generation using Variational Autoencoders

Kutay Bölat

TU Delft

Contents

- Generative modelling (reminder)
- Latent variable models
- Variational autoencoders
- Hands-on session

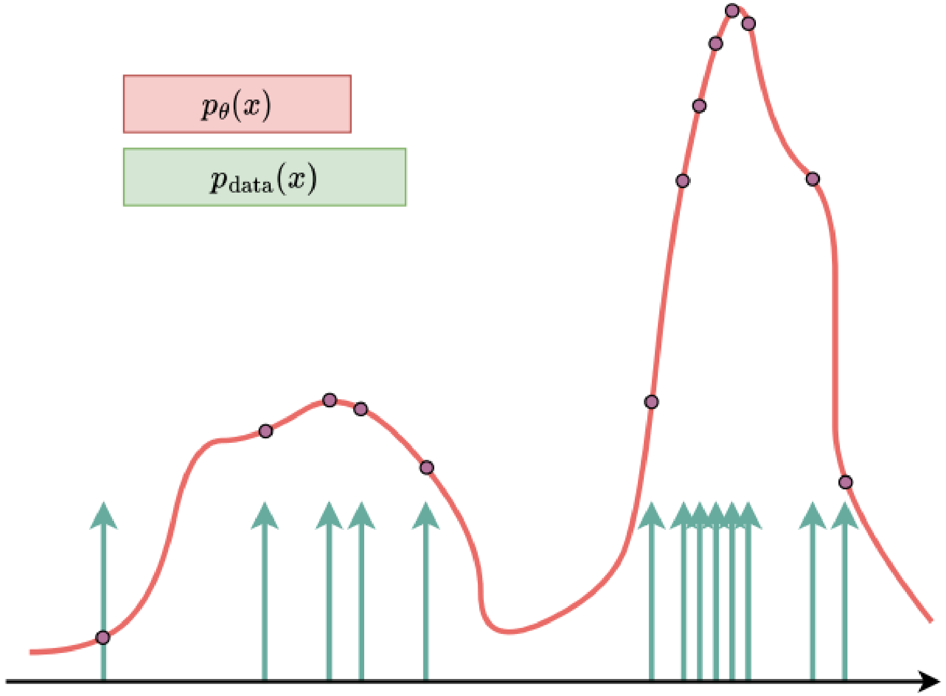


Generative modelling (reminder)

Generative modelling (reminder)

$$p_{\text{real-life}}(\mathbf{x}) \to p_{\text{data}}(\mathbf{x}) = \frac{1}{N}\sum_{\mathcal{X}} \delta(\mathbf{x} - \mathbf{x}_i)$$

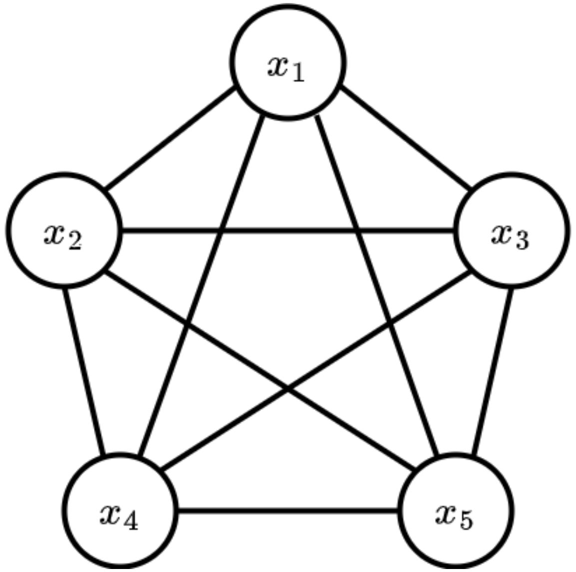
$$\underset{\theta}{\text{argmin}}\ D_{KL}(p_{\text{data}}(\mathbf{x}) || p_{\theta}(\mathbf{x})) = \underset{\theta}{\text{argmax}}\ \frac{1}{N}\sum_{\mathcal{X}} \log p_{\theta}(\mathbf{x}_i)$$



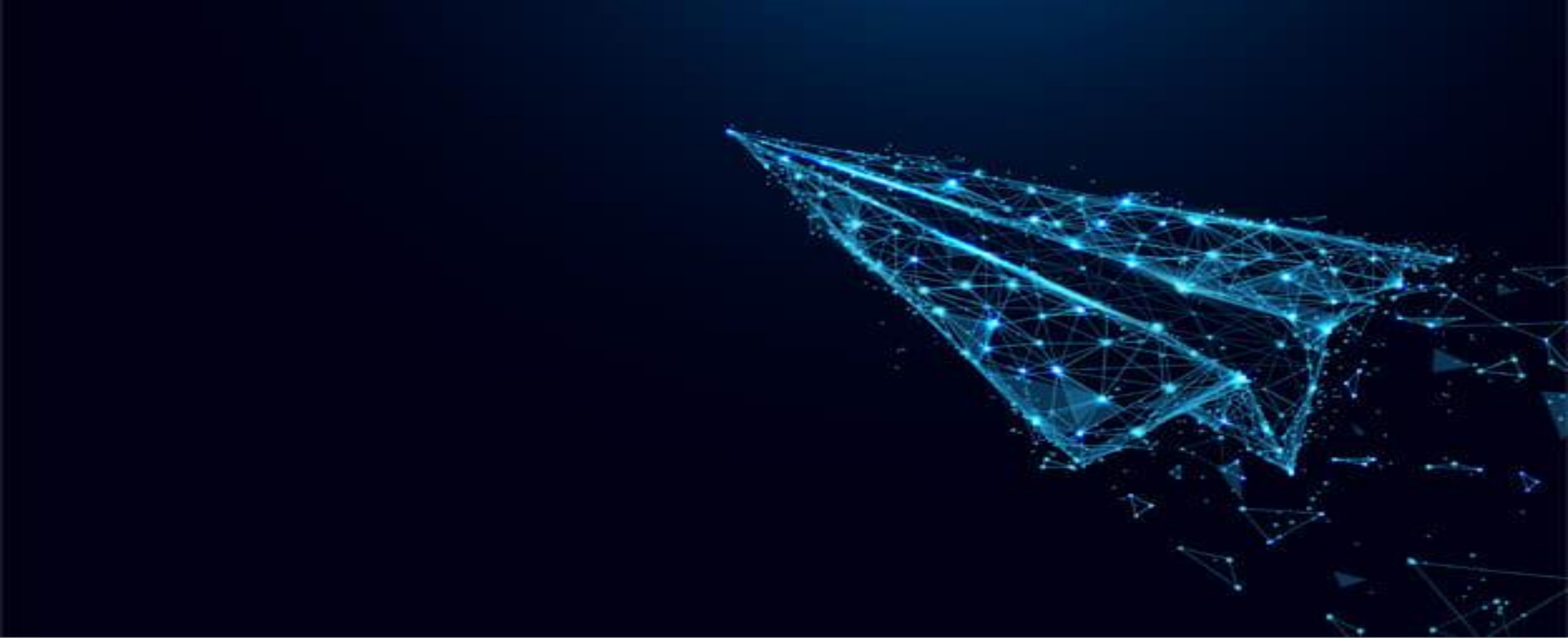
Generative modelling (reminder)

Goal: Find a (parameterised) probabilistic model $p(\boldsymbol{x})$, where x is high-dimensional.

Problem: Finding/learning relations between many features is exceedingly hard (even for very deep and wide neural networks).



$$p(\boldsymbol{x}) = p(x_1, x_2, x_3, \dots x_d)$$
$$= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \dots p(x_d|x_1, \dots, x_{d-1})$$

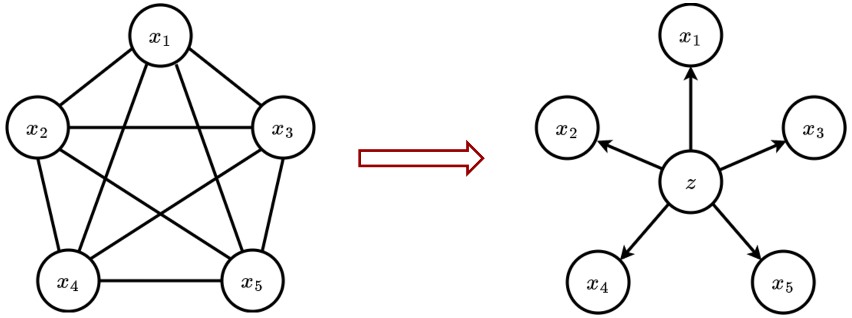


Latent variable models

Latent variable models

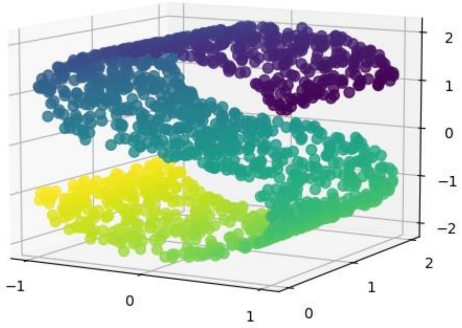
Idea: Introduce unobserved auxiliary variables (latent variables)

- Observables: $\boldsymbol{x} \in \mathbb{R}^d$
- Latent variables: $\mathbf{z} \in \mathbb{R}^m$



Simplified model structure:

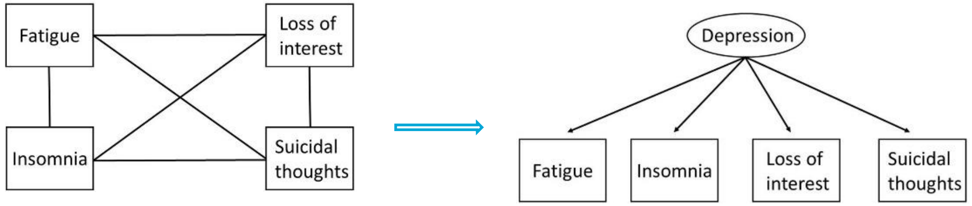
- $p(\boldsymbol{x}) = \int \; p(\boldsymbol{x}, \mathrm{z}) dz = \int \; p(z) \prod_{j=1}^{d} p\left(x_j \middle| z\right) dz$



scikit-learn.org/stable/auto_examples/manifold/plot_compare_methods.html

Motivation:

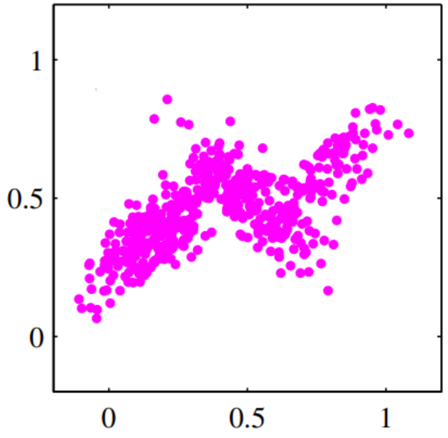
- 'Meaningful' data tends to reside on a lower dimensional manifold in high-dimensional space.
- Often, known causal relations can simplify the model.



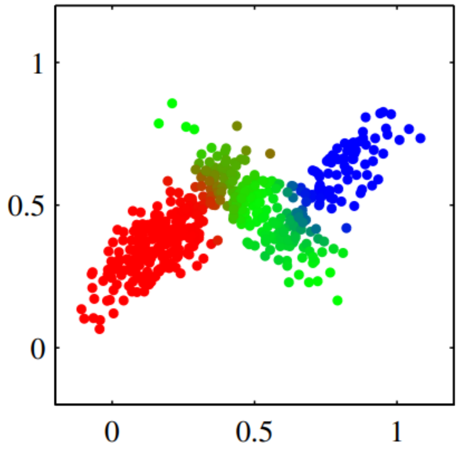
https://www.frontiersin.org/articles/10.3389/fpsyg.2017.00798/full

Latent variable models

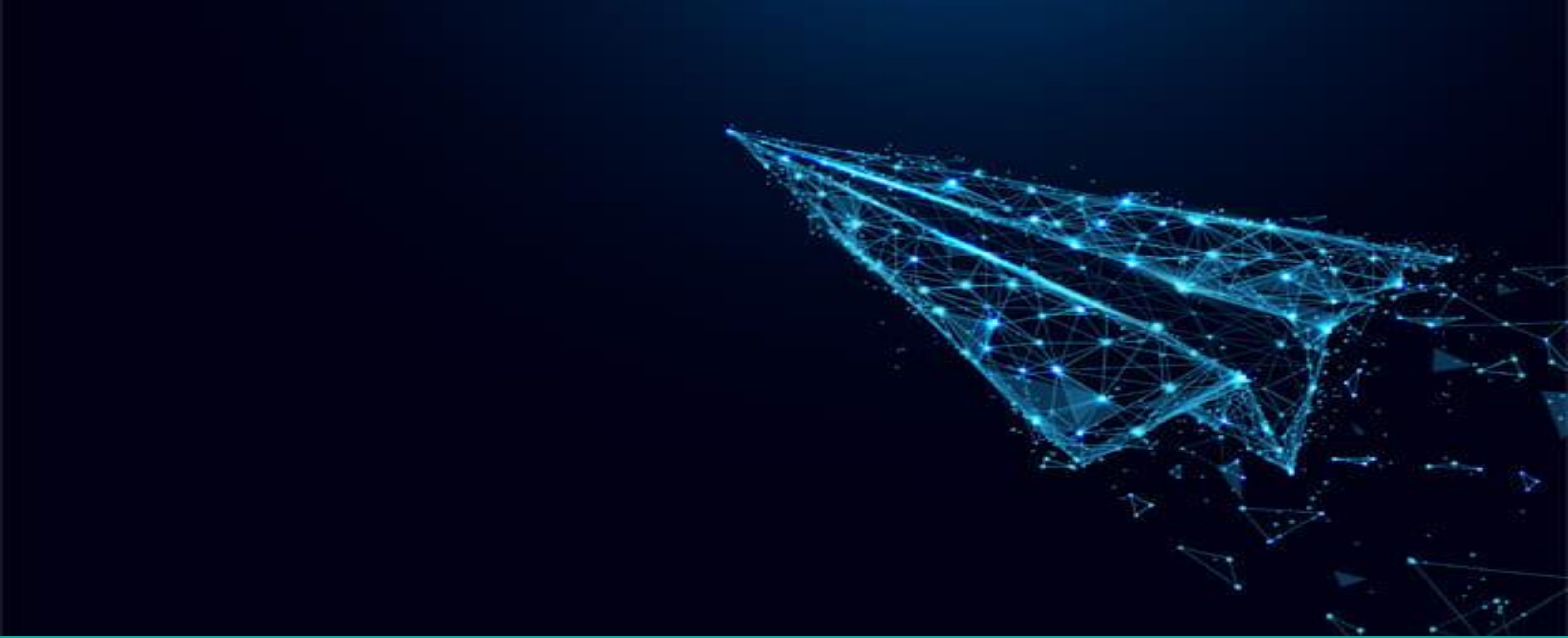
- Gaussian mixture models



Requires $p(\boldsymbol{x})$ to be expressive enough to capture non-linearities.



$p(\boldsymbol{x}) = \sum_{z} p(\boldsymbol{x}|z)p(z) = \sum_{z\in\{0,1,2\}} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)p(z)$

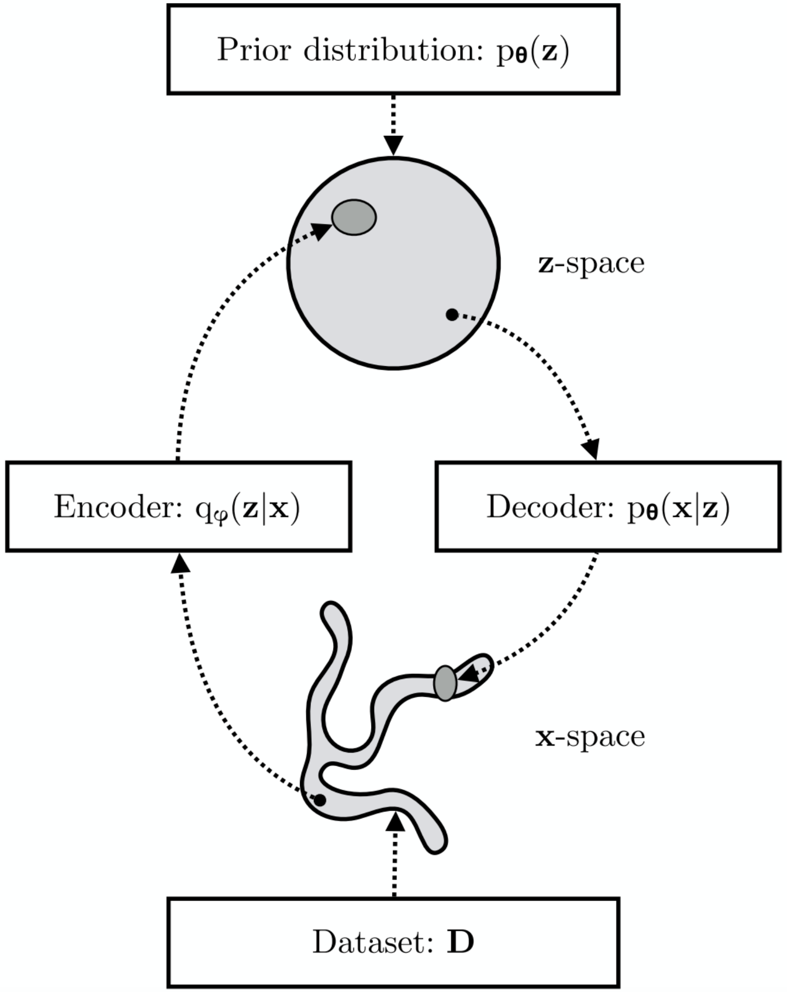


Variational autoencoders

Variational autoencoders - Introduction

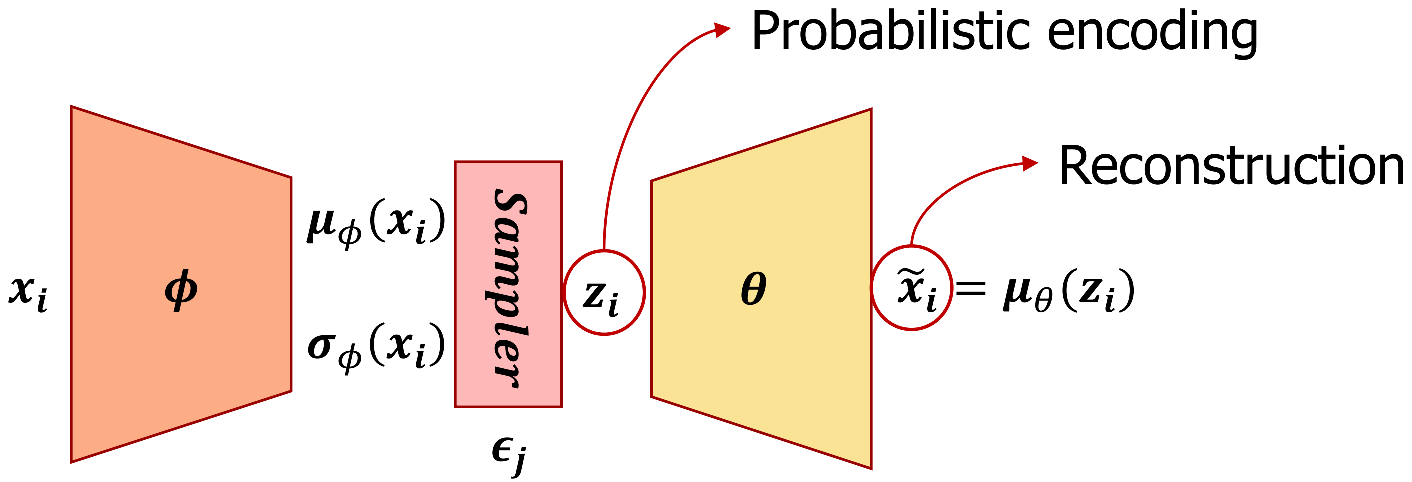
- Introduced in Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*
  - *Over 37000 citations!*

- Two interpretations:
  - A probabilistic autoencoder
  - Latent variable models parametrized by NNs



Kingma, D. P., & Welling, M. (2019). An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, *12*(4), 307–392.

Variational autoencoders – End-result



Intuitive objective:
1. Good reconstruction: $\tilde{\boldsymbol{x}}_i \approx \boldsymbol{x}_i$ ($\boldsymbol{z}_i$ holds important information about $\boldsymbol{x}_i$)
2. Regularization: $p(\boldsymbol{z}_i|\boldsymbol{x}_i) \approx N(0, I)$ ($\boldsymbol{z}_i$ holds important information about other data points)

$$\mathcal{L}(\phi,\theta,X) = \underbrace{-\frac{1}{NM}\sum_{i,j}^{n,M}\left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_\theta(g_\phi(\boldsymbol{x}^{(i)}, \boldsymbol{\epsilon}_j^{(i)}))\right\|_2^2}_{\text{reconstruction error}} \underbrace{-\frac{1}{2}\sum_{k=1}^{m}\left(\mu_{k_\phi}(\boldsymbol{x}^{(i)})\right)^2 + \left(\sigma_{k_\phi}(\boldsymbol{x}^{(i)})\right)^2 - 1 - 2\log\sigma_{k_\phi}(\boldsymbol{x}^{(i)})}_{\text{regularization}}$$

Latent variable models - Math

- With $p_\theta(\boldsymbol{z})$ and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ we can generate data!
- … but learning $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ requires $p_\theta(\boldsymbol{z}|\boldsymbol{x})$
- We will use an **approximate posterior** $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ (a separate model)

Definitions

- The model is $p_\theta(\boldsymbol{x})$ where $\int_{\mathcal{X}} p_\theta(\boldsymbol{x})d\boldsymbol{x} = 1$ and $p_\theta(\boldsymbol{x}) > 0$.
- $p_\theta(\boldsymbol{x}) = \int_{\mathcal{Z}} p_\theta(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{z} = \int_{\mathcal{Z}} p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\theta(\boldsymbol{z})\, d\boldsymbol{z}$.

Some useful operations

- $p_\theta(\boldsymbol{x}|\boldsymbol{z})$: Called the **likelihood** or the **inference** model. Very flexible. Can be modelled with a neural network.
- $p_\theta(\boldsymbol{z})$: Called the **prior**. Can be modelled with simple distributions and it transfers prior assumptions to the model.
- $\int_{\mathcal{Z}} \dots d\boldsymbol{z}$: Averaging over latent variables. **Marginalization**. Generally **intractable**, i.e. impossible to compute analytically and extremely expensive to compute numerically.

- Also, we might want to infer (low dimensional) latent variables by looking at (high dimensional) data, i.e. **compression**. This requires us to compute the **posterior**

$$p_\theta(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{x})} = \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\theta(\boldsymbol{z})}{\int_{\mathcal{Z}} p_\theta(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{z}}$$

which is also intractable most of the time.

Variational autoencoders – Objective

- Maximum log-likelihood criteria: $\underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{p_{data}(x)} \log p_\theta(\boldsymbol{x}) = \underset{\theta}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}_i)$
- $\log p_\theta(\boldsymbol{x}) = \log\left(\frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\theta(\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right) = \log\left(\frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\theta(\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})} \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right) = \log(p_\theta(\boldsymbol{x}|\boldsymbol{z})) - \log\left(\frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z})}\right) + \log\left(\frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right)$
- $\log p_\theta(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log(p_\theta(\boldsymbol{x}|\boldsymbol{z})) - \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log\left(\frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z})}\right) + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log\left(\frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right)$

  $= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log(p_\theta(\boldsymbol{x}|\boldsymbol{z})) - D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})\right) + D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})\right)$

  - $\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \log(p_\theta(\boldsymbol{x}|\boldsymbol{z}))$: Reconstruction log-likelihood
  - $D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})\right)$: KL-divergence (discrepancy) between the approx. posterior and the prior
  - $D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})\right)$: KL-divergence (discrepancy) between the approx. posterior and the true posterior

Variational autoencoders – Objective

$$\log p_\theta(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{x})} \log(p_\theta(\boldsymbol{x}|\mathbf{z})) - D_{KL}\left(q_\phi(\mathbf{z}|\boldsymbol{x}) \parallel p_\theta(\mathbf{z})\right) + D_{KL}\left(q_\phi(\mathbf{z}|\boldsymbol{x}) \parallel p_\theta(\mathbf{z}|\boldsymbol{x})\right)$$

- Remarks:
  - KL-divergence is always positively defined.
  - The only intractable term is $D_{KL}\left(q_\phi(\mathbf{z}|\boldsymbol{x}) \parallel p_\theta(\mathbf{z}|\boldsymbol{x})\right)$.
- $\log p_\theta(\boldsymbol{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{x})} \log(p_\theta(\boldsymbol{x}|\mathbf{z})) - D_{KL}\left(q_\phi(\mathbf{z}|\boldsymbol{x}) \parallel p_\theta(\mathbf{z})\right) = ELBO$
- Objective:

$$\phi^*, \theta^* = \underset{\phi,\theta}{\operatorname{argmax}}\, \mathbb{E}_{p_{data}(\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{x})} \log(p_\theta(\boldsymbol{x}|\mathbf{z})) - D_{KL}\left(q_\phi(\mathbf{z}|\boldsymbol{x}) \parallel p_\theta(\mathbf{z})\right)\right]$$

reconstruction success KL-divergence

Variational autoencoders - Recipe

- How to implement and train VAEs?

- **Step 1:** Selection of distributions
  - $p_\theta(\boldsymbol{x}|\boldsymbol{z})$: Totally depends on the application. It can be:
    - $p_\theta(\boldsymbol{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_\theta(\boldsymbol{z}), \boldsymbol{\Sigma}_\theta(\boldsymbol{z}))$ if $\mathcal{X} = \mathbb{R}^d$
    - $p_\theta(\boldsymbol{x}|\mathbf{z}) = Lognormal(\boldsymbol{x}|\boldsymbol{\mu}_\theta(\mathbf{z}), \boldsymbol{\Sigma}_\theta(\mathbf{z}))$ if $\mathcal{X} = \mathbb{R^+}^d$
    - $p_\theta(\boldsymbol{x}|\mathbf{z}) = Beta(\boldsymbol{x}|\boldsymbol{\beta}_\theta(\mathbf{z}), \boldsymbol{\alpha}_\theta(\mathbf{z}))$ if $\mathcal{X} = [0,1]^d$
    - $p_\theta(\boldsymbol{x}|\mathbf{z}) = Categorical(\boldsymbol{x}|\boldsymbol{P}_\theta(\mathbf{z}))$ if $\mathcal{X} = \{0,1,\dots,K\}^d$

  - $q_\phi(\boldsymbol{z}|\boldsymbol{x})$: Should result in an analytical $D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})\right)$ expression. Also, we must be able to take samples in a "straightforward' manner. (More on this later)

  - $p_\theta(\boldsymbol{z})$: Should result in an analytical $D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})\right)$ expression.

Variational autoencoders - Recipe

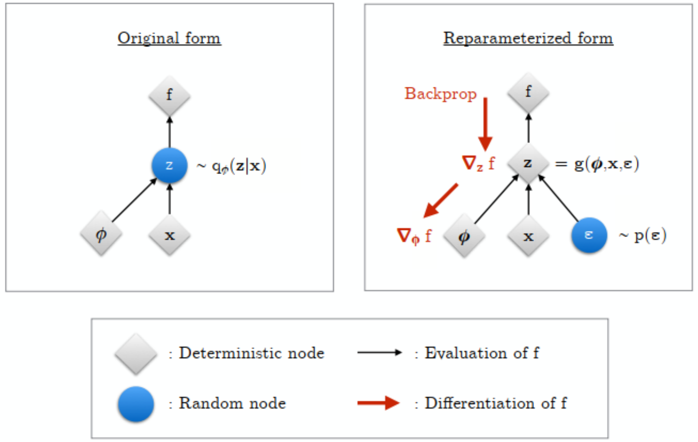
- Investigating the ELBO – **Reconstruction log-likelihood**
  - $\mathbb{E}_{p_{data}(x)}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\log(p_\theta(\boldsymbol{x}|\boldsymbol{z}))\right] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{q_\phi\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right)}\log\left(p_\theta\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right)\right)$ (No closed form expression)

    $= \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\log\left(p_\theta\left(\boldsymbol{x}^{(i)}\middle|\boldsymbol{z}_j^{(i)}\right)\right)$, where $\boldsymbol{z}_j^{(i)} \sim q_\phi\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right)$ (Monte Carlo approximation)

- **Problem**: Sampling is not a differentiable operation.

- **Solution**: Reparameterization trick (**Step 2**)
  - $\boldsymbol{z}_j^{(i)} \sim q_\phi\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right) \rightarrow \boldsymbol{z}_j^{(i)} = g_\phi(\boldsymbol{x}^{(i)}, \boldsymbol{\epsilon}_j^{(i)})$, where $\boldsymbol{\epsilon}_j^{(i)} \sim p(\boldsymbol{\epsilon})$
  - This separation should be “straightforward”, i.e. computationally cheap.

- For $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}_\phi(\boldsymbol{x}), diag(\boldsymbol{\sigma}_\phi(\boldsymbol{x}))\right)$
  - $\boldsymbol{z}_j^{(i)} = \boldsymbol{\mu}_\phi\left(\boldsymbol{x}^{(i)}\right) + \boldsymbol{\sigma}_\phi\left(\boldsymbol{x}^{(i)}\right) \odot \boldsymbol{\epsilon}_j^{(i)}$, where $\boldsymbol{\epsilon}_j^{(i)} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$



Kingma, Diederik P., and Max Welling. “An Introduction to Variational Autoencoders.” Foundations and Trends® in Machine Learning, vol. 12, no. 4, 2019, pp. 307–92. Crossref, https://doi.org/10.1561/2200000056.

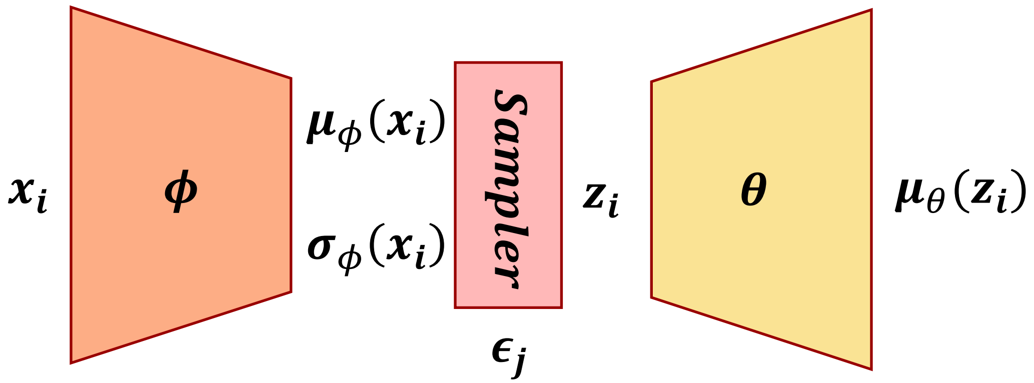
Variational autoencoders - Recipe

- Investigating the ELBO – **KL divergence**
  - **Step 3:** Analytically derive the KL divergence term
  - Let's keep $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}_\phi(\boldsymbol{x}), diag(\boldsymbol{\sigma}_\phi(\boldsymbol{x}))\right)$ and $p_\theta(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$
    - $D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})\right) = \frac{1}{2}\sum_{k=1}^{m}\left(\mu_{k_\phi}(\boldsymbol{x})\right)^2 + \left(\sigma_{k_\phi}(\boldsymbol{x})\right)^2 - 1 - 2\log\sigma_{k_\phi}(\boldsymbol{x})$

  - **Step 3.5:** Put everything together
  - For $p_\theta\left(\boldsymbol{x}^{(i)}\middle|g_\phi(\boldsymbol{x}^{(i)}, \boldsymbol{\epsilon}_j^{(i)})\right) = \mathcal{N}\left(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}_\theta(g_\phi(\boldsymbol{x}^{(i)}, \boldsymbol{\epsilon}_j^{(i)})), \mathbf{I}\right)$, ELBO becomes
    - $\mathcal{L}(\phi,\theta,X) = \underbrace{-\frac{1}{NM}\sum_{i,j}^{n,M}\left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_\theta(g_\phi(\boldsymbol{x}^{(i)}, \boldsymbol{\epsilon}_j^{(i)}))\right\|_2^2}_{\text{reconstruction error}} \underbrace{- \frac{1}{2}\sum_{k=1}^{m}\left(\mu_{k_\phi}(\boldsymbol{x}^{(i)})\right)^2 + \left(\sigma_{k_\phi}(\boldsymbol{x}^{(i)})\right)^2 - 1 - 2\log\sigma_{k_\phi}(\boldsymbol{x}^{(i)})}_{\text{KL-divergence}}$

Variational autoencoders - Recipe

- **Step 4:** Replace all the parameterized functions with neural networks



$$\mathcal{L}(\phi,\theta,X) = -\frac{1}{NM}\sum_{i,j}^{n,M} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}_\theta(g_\phi(\boldsymbol{x}^{(i)},\boldsymbol{\epsilon}_j^{(i)})) \right\|_2^2 - \frac{1}{2}\sum_{k=1}^{m} \left(\mu_{k_\phi}(\boldsymbol{x}^{(i)})\right)^2 + \left(\sigma_{k_\phi}(\boldsymbol{x}^{(i)})\right)^2 - 1 - 2\log\sigma_{k_\phi}(\boldsymbol{x}^{(i)})$$

- **Step 5:** Let your favourite DL library (PyTorch, Tensorflow) does its magic.

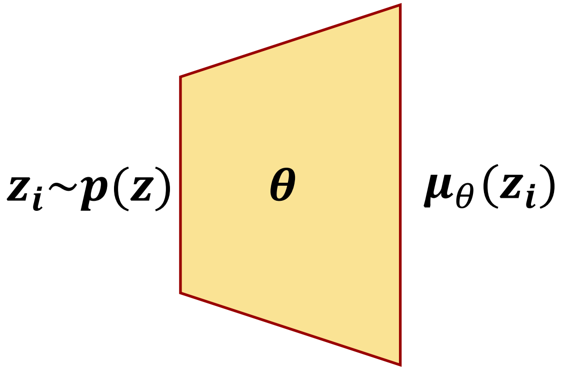
Variational autoencoders - Recipe

- Investigating the ELBO – **Overall**
  - $\mathcal{L}(\phi,\theta,X) = -\frac{1}{NM}\sum_{i,j}^{n,M}\left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_\theta\left(g_\phi\left(\boldsymbol{x}^{(i)},\boldsymbol{\epsilon}_j^{(i)}\right)\right)\right\|_2^2 - \frac{1}{2}\sum_{k=1}^{m}\left(\mu_{k_\phi}\left(\boldsymbol{x}^{(i)}\right)\right)^2 + \left(\sigma_{k_\phi}\left(\boldsymbol{x}^{(i)}\right)\right)^2 - 1 - 2\log\sigma_{k_\phi}\left(\boldsymbol{x}^{(i)}\right)$
  - $-\mathcal{L}(\phi,\theta,X)$ is pretty much identical to "**Autoencoder/Reconstruction Loss**" + "**KL regularization**“

  - Maximizing ELBO (minimizing $-\mathcal{L}(\phi,\theta,X)$) encourages
    - Inputs and outputs to be the same as much as possible
    - $\mu_{k_\phi}\left(\boldsymbol{x}^{(i)}\right)$s to be close to 0 as much as possible
    - $\sigma_{k_\phi}\left(\boldsymbol{x}^{(i)}\right)$s to be close to 1 as much as possible

Variational autoencoders - Recipe

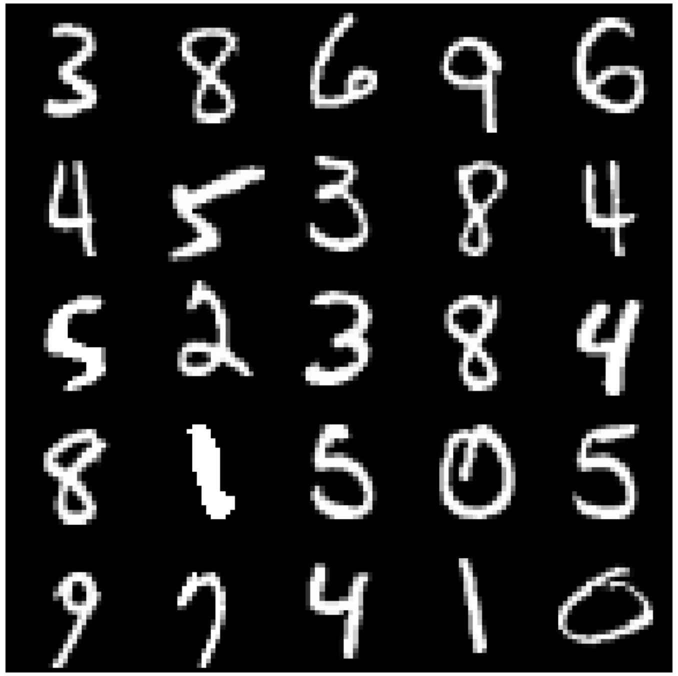
- **An important remark**
  - We use $p(z)$ for two reasons in practice:
    1. Regularizing the latent encodings **during training**.
    2. Generating new data **during inference**.



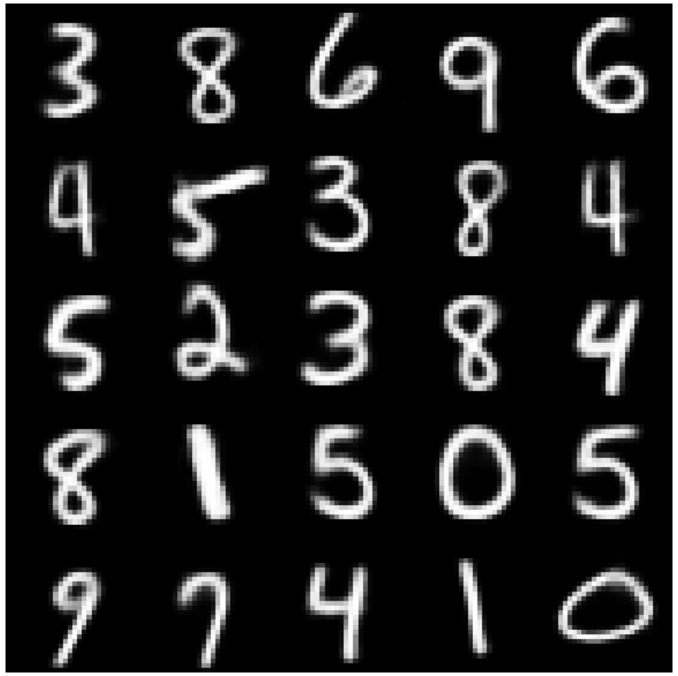
- For generating synthetic data, we DO NOT have any access to original data or their latent encodings!

Variational autoencoders - Examples

- Image data (MNIST)



Original

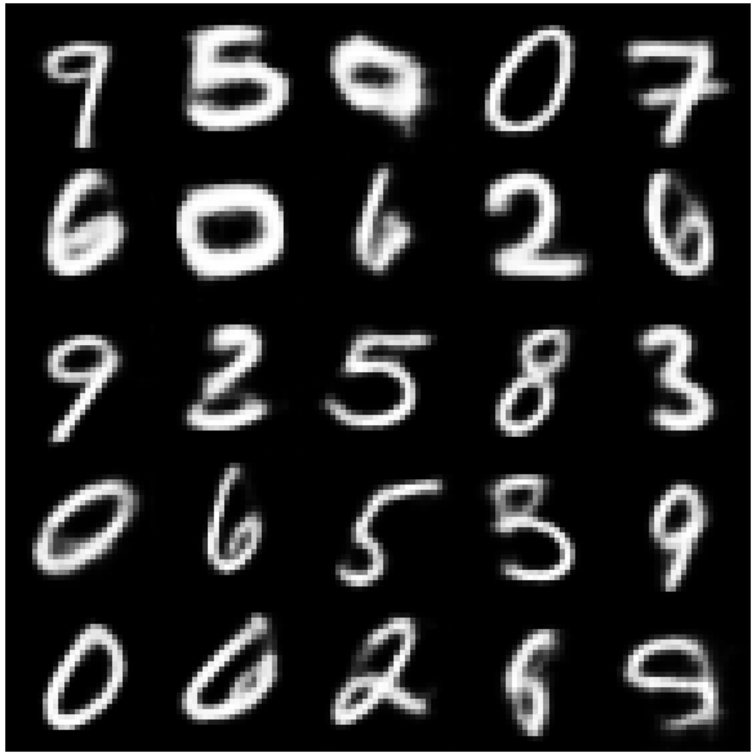


Reconstruction

github.com/clementchadebec/benchmark_VAE

Variational autoencoders - Examples

- Image data (MNIST)



Samples

github.com/clementchadebec/benchmark_VAE

Variational autoencoders - Examples

- Image data (CelebA)

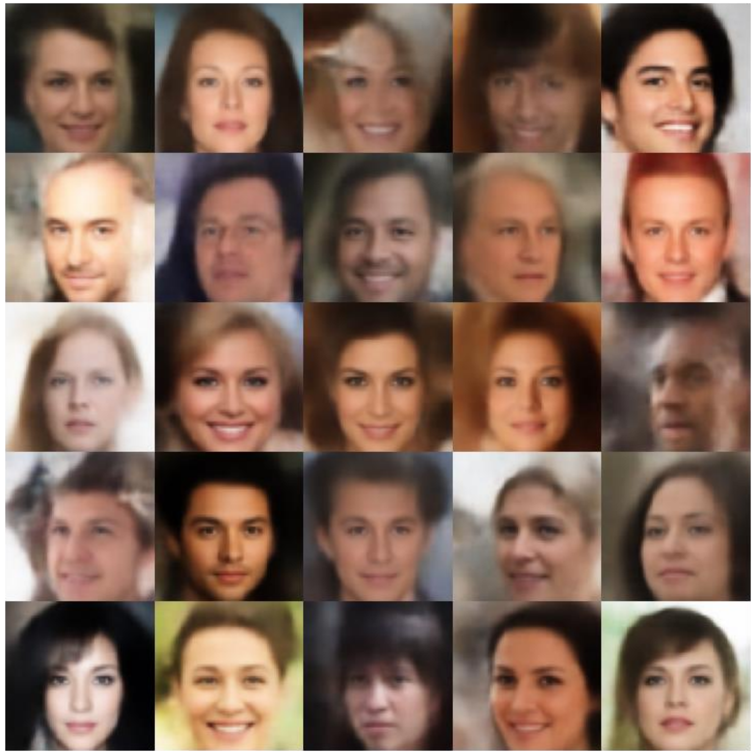
Original

Reconstruction

github.com/clementchadebec/benchmark_VAE

Variational autoencoders - Examples

- Image data (CelebA)

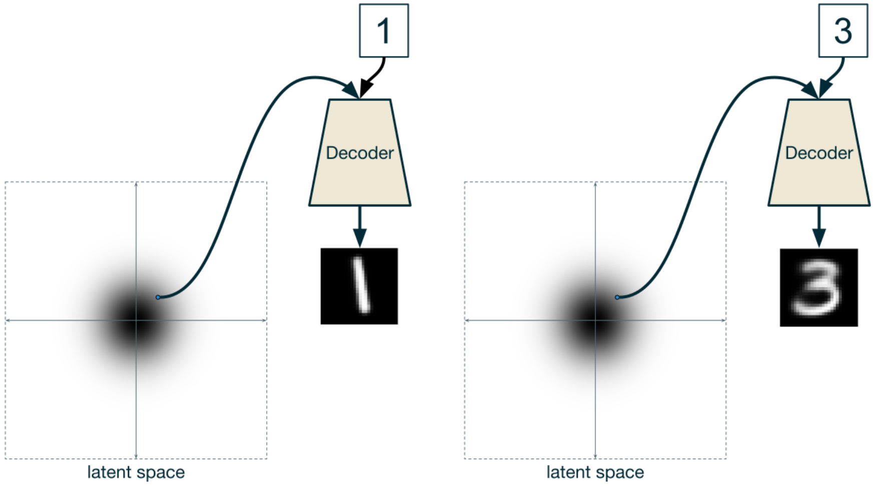


Samples

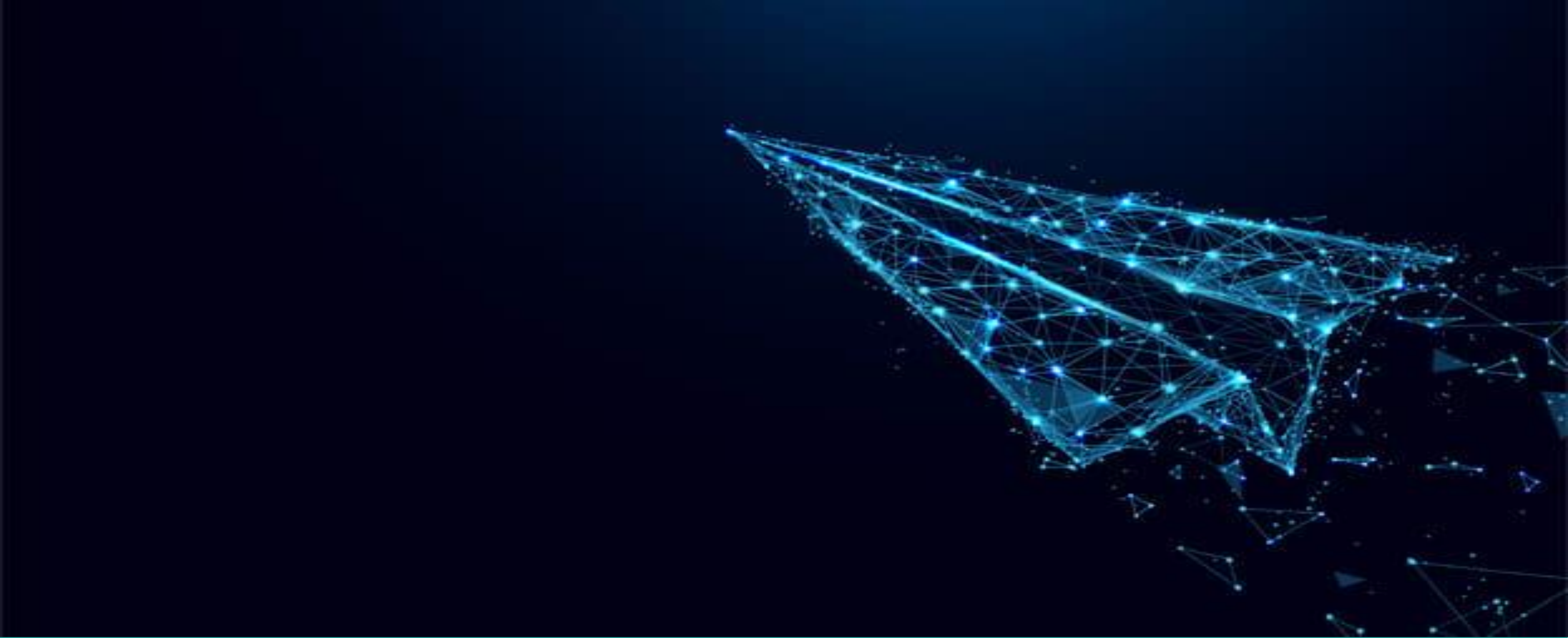
github.com/clementchadebec/benchmark_VAE

Variational autoencoders - Extensions

- Conditional VAE
  - $p_\theta(\boldsymbol{x}) \longrightarrow p_\theta(\boldsymbol{x}|\boldsymbol{y})$
  - ELBO: $\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})} \log\big(p_\theta(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{y})\big) - D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) \parallel p_\theta(\boldsymbol{z},\boldsymbol{y})\right)$



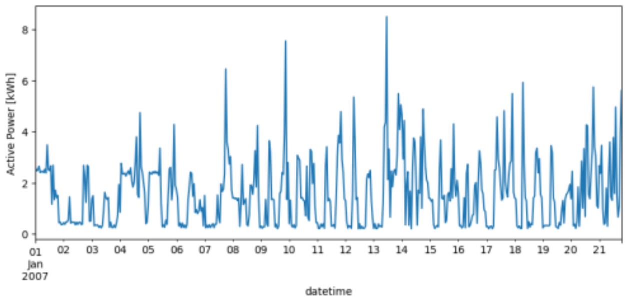
ijdykeman.github.io/ml/2016/12/21/cvae.html

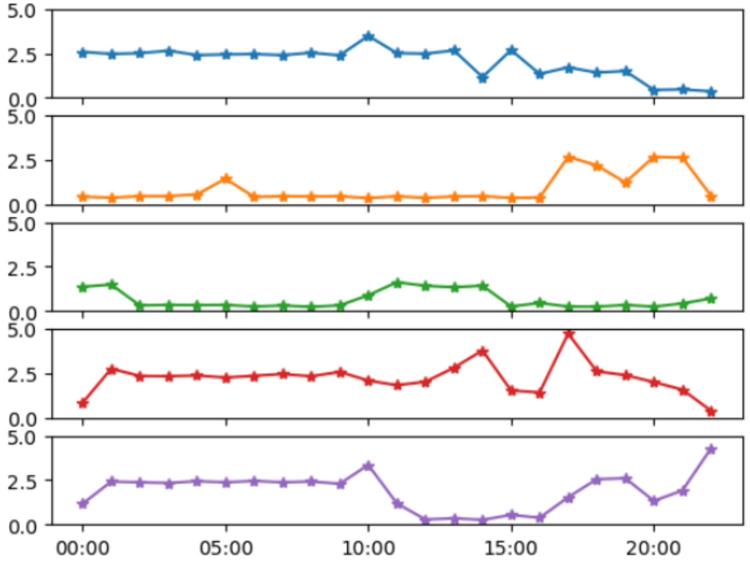


Hands-on session

Hands-on session

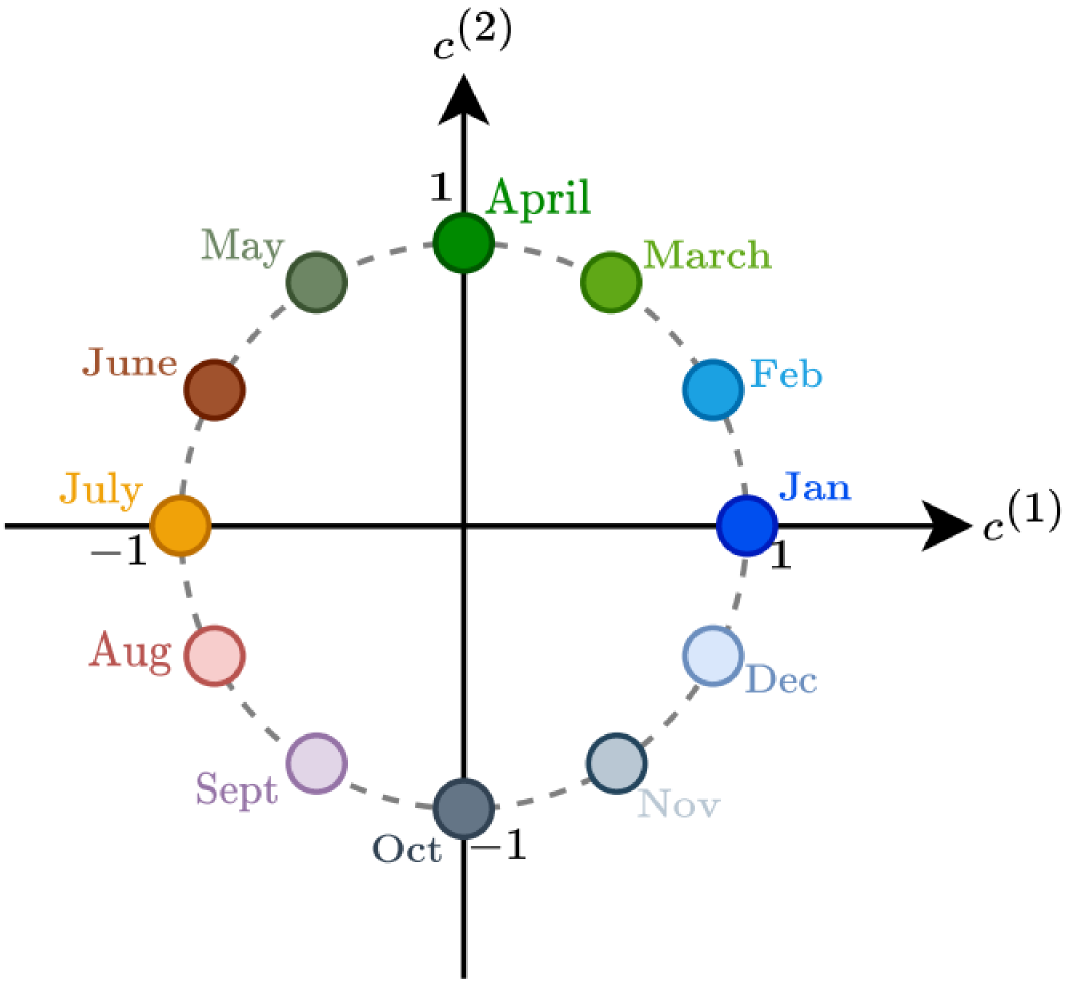
- Data representation
  - Daily snapshots



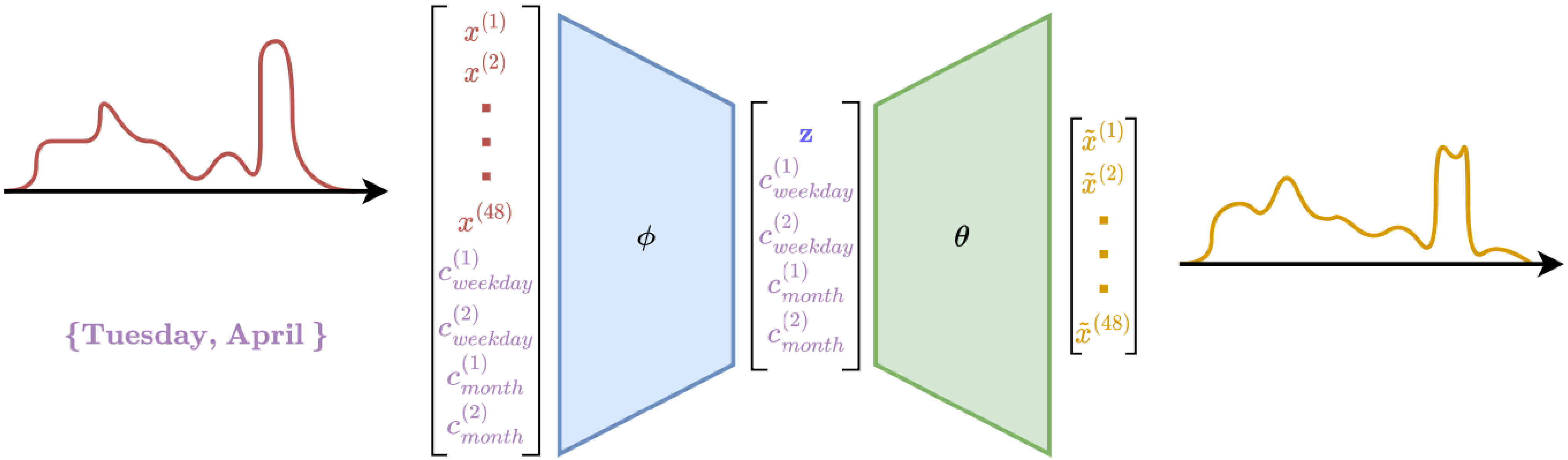


Hands-on session

- Conditions
  - Months and days of the week
  - Circular transform



Hands-on session



Hands-on session

- github.com/kabolat/innocypes-summer-school_synthetic-data-tutorial

Thanks for your attention!

Any questions?

Kutay Bölat

K.Bolat@tudelft.nl
github.com/kabolat

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956433.