~~Défi 3 : AI for time series forecasting~~

Défi 3 : Generative Artificial Intelligence (I/III)

Défis en Intelligence Artificielle

Souhaib Ben Taieb & Victor Dheur

7 Décembre 2023

Université de Mons

Teaching assistant

Victor DHEUR

PhD candidate

Big Data and Machine Learning Lab

Department of Computer Science

victor.dheur@umons.ac.be

Schedule

- *Week 1: 7 December 2023, 6-9 pm*
  - Introduction to generative modelling
- *Week 2: 14 December 2023, 6-9 pm*
  - (Subject to change) Generative Adversarial Networks (GANs)
- *Week 3: 21 December 2023, 6-9 pm*
  - (Subject to change) Diffusion models
- *Project report: 17 January 2023, 11:55 pm*
  - More details TBA soon

Communication

- Course webpage
  - `https://github.com/bsouhaib/Hands-On-AI-2023-Challenge3`
  - Lecture slides
  - Labs
- Moodle
  - `https://moodle.umons.ac.be/course/view.php?id=2666#section-4`
  - Forum for asking questions
  - Project submission
- **No email please — use the forum**

Introduction

Different types of learning problems

- Supervised learning (Défi 1)
  - Data: (*input*, *output*) or (*input*, *label*)
  - Goal: Learn function to map input to output
  - Examples: Classification, regression, object detection, etc.
- Reinforcement learning (Défi 2)
  - Data: (input, some output, grade for this output) or (state, action, reward)
  - Goal: Learn to take actions in an environment to maximize a cumulative reward

Different types of learning problems

- Supervised learning (Défi 1)
  - Data: (*input*, *output*) or (*input*, *label*)
  - Goal: Learn function to map input to output
  - Examples: Classification, regression, object detection, etc.
- Reinforcement learning (Défi 2)
  - Data: (input, some output, grade for this output) or (state, action, reward)
  - Goal: Learn to take actions in an environment to maximize a cumulative reward
- **Unsupervised learning (Défi 3)**
  - Data: (input)
  - Goal: Learn the (hidden) underlying structure of the data
  - Examples: Clustering, dimensionality reduction, **generative modelling**, etc.

Generative modeling

Goal: Take as input training samples from some distribution and learn a model that represents that distribution

Density Estimation

samples

Sample Generation

Input samples

Training data ~ $P_{data}(x)$

Generated samples

Generated ~ $P_{model}(x)$

How can we learn $P_{model}(x)$ similar to $P_{data}(x)$?

Why generative models? Debiasing

Capable of uncovering **underlying features** in a dataset

Homogeneous skin color, pose

VS

Diverse skin color, pose, illumination

How can we use this information to create fair and representative datasets?

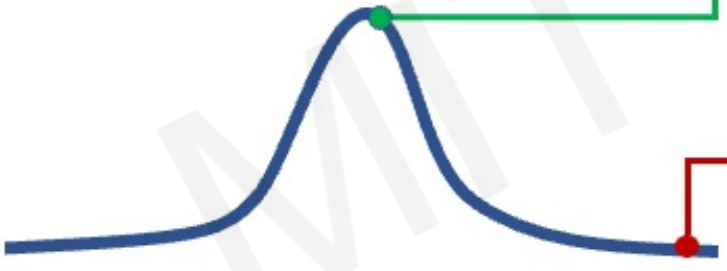
Software Lab!

Why generative models? Outlier detection

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!

95% of Driving Data:
(1) sunny, (2) highway, (3) straight road

Detect outliers to avoid unpredictable behavior when training



Edge Cases

Harsh Weather

Pedestrians

Which face is real?

A

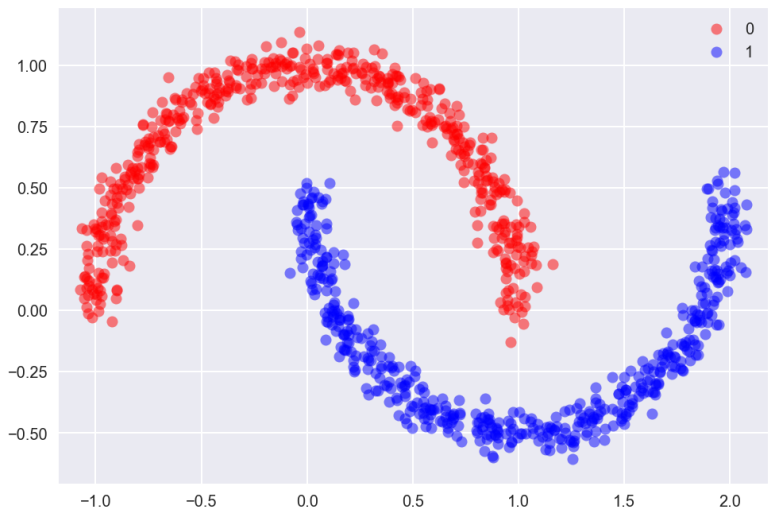
B

C

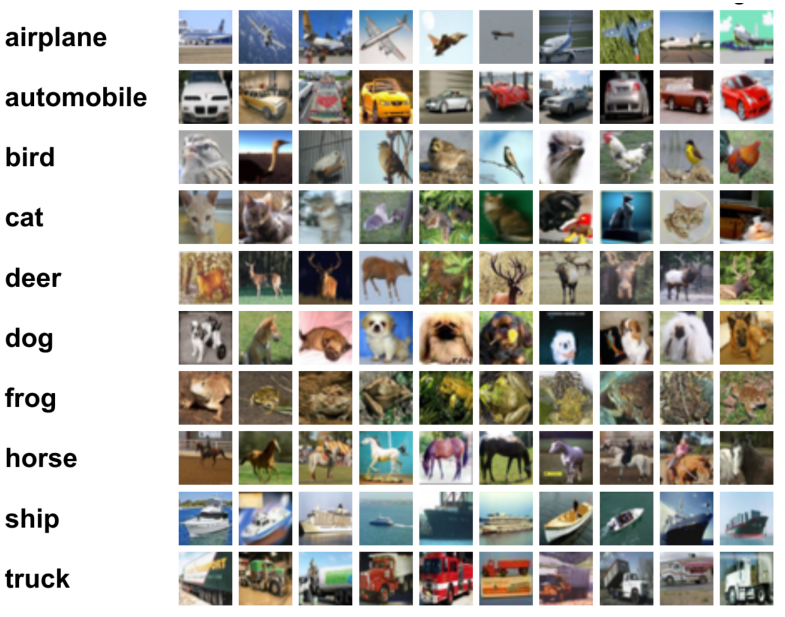
Generative models

A **generative model** is a statistical model of the **probability distribution** $p(\boldsymbol{x})$ over a domain $\mathcal{X} \subseteq \mathbb{R}^D$.

If it is conditioned on inputs $\boldsymbol{c} \in \mathcal{C}$, it becomes a **conditional generative model** $p(\boldsymbol{x} \mid \boldsymbol{c})$.



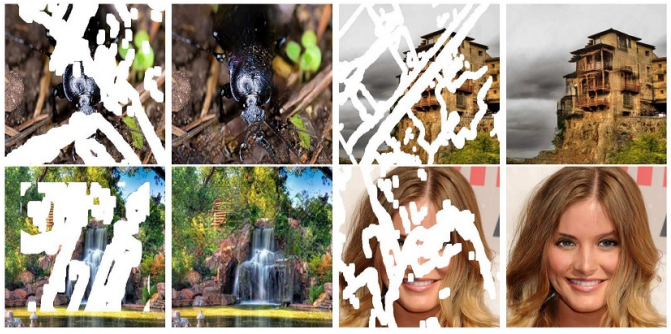
(a) Two moons dataset ($\mathcal{D} = 2$)



(b) CIFAR-10 dataset ($\mathcal{D} = 32 \times 32 \times 3$)

Examples of data generation applications

c	x	Examples of applications
Text	Text	Conversational bots (e.g. ChatGPT), translation, summarization
Text	Image	Digital creation assistant tools
Image	Text	Image search engines, captioning
Image	Image	Inpainting, colorization, style transfer
Text	Speech	Voice assistants
Protein sequence	3D structure	Drug discovery



(a) Inpainting (Source: [Liu+18])

(b) Colorization (Source: [Ant])

Examples of data generation applications

V Can you give me a structure for a lecture about deep generative models? I would talk about maximum likelihood estimation, autoregressive model, normalizing flows and variational autoencoders.

Sure, here's a possible structure for a lecture on deep generative models, covering maximum likelihood estimation, autoregressive models, normalizing flows, and variational autoencoders:

I. Introduction to Deep Generative Models

- Definition and motivation
- Applications and examples

II. Maximum Likelihood Estimation

- Definition and formulation
- Optimization using gradient descent
- Issues with high-dimensional data and solutions

III. Autoregressive Models

- Definition and intuition
- Example models: PixelCNN

Stop generating

(a) Text-to-text sample (ChatGPT)

(b) Text-to-image sample (Midjourney V4). Prompt: `Belgian super hero black and red and yellow black wings beer lion powers full body.`

Examples of data generation applications

A quick statistics recap

Discrete Random Variables

A discrete random variable $X \in \mathcal{X} \subseteq \mathbb{R}$ can be characterized by its **probability mass function** (PMF), a function which specifies the probability of x, i.e.

$$p_X(x) = \mathbb{P}(X = x),$$

where $p_X(x) \geq 0, \forall x \in \mathcal{X}$ and $\sum_{x \in \mathcal{X}} p_X(x) = 1$.

An example with three possible outcomes:

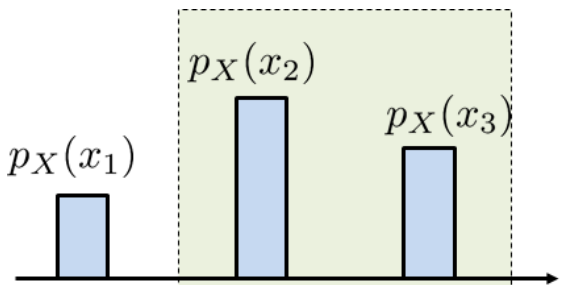


Image source: Introduction to Probability for Data Science, Stanley H. Chan.

Discrete Random Variables

The **expectation** of a discrete random variable X is

$$\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x \; p_X(x).$$

The **variance** of a discrete random variable X is

$$\mathrm{Var}(X) = \mathbb{E}[(X - \mu_X)^2],$$

where $\mu_X = \mathbb{E}[X]$. The **standard deviation** of X is $\sqrt{\mathrm{Var}(X)}$.

Some important discrete distributions

- A **uniform** distribution on K categories. The PMF of $X \in \{C_1, C_2, \ldots, C_K\}$ is given by

$$p_X(x) = \frac{1}{K}, \qquad \forall x \in \{C_1, C_2, \ldots, C_K\}$$

- The **Bernouilli** distribution with parameter $p \in [0, 1]$. The PMF of $X \in \{0, 1\}$ is given by

$$p_X(x) = p^x(1-p)^{1-x}$$

- Other important distributions: Binomial, Geometric, Poisson, etc.
- The symbol "$\sim$" denotes "distributed as", i.e. $X \sim \text{Ber}(p)$ means that X has a Bernoulli distribution with parameter p.

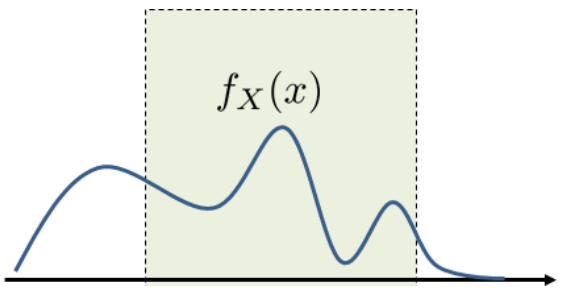
Continuous Random Variables

A continuous random variable $X \in \mathcal{X} \subseteq \mathbb{R}$ can be described by its **probability density function** (PDF), a function p_X, when integrated over an interval $[a, b]$, yields the probability of obtaining $a \leq X \leq b$:

$$\mathbb{P}(a \leq X \leq b) = \int_a^b p_X(x)dx.$$

A PDF has the following properties:

1. $p_X(x) \geq 0, \forall x \in \mathcal{X}$
2. $\int_{\mathcal{X}} p_X(x)\ dx = 1$



Continuous Random Variables

The **expectation** of a continuous random variable X is given by

$$\mathbb{E}[X] = \int_{\mathcal{X}} x \; p_X(x) \; dx.$$

The **variance** of a continuous random variable X is

$$\text{Var}(X) = \mathbb{E}[(X - \mu_X)^2] = \int_{\mathcal{X}} (x - \mu_X)^2 \; p_X(x) dx,$$

where $\mu_X = \mathbb{E}[X]$. The **standard deviation** of X is $\sqrt{\text{Var}(X)}$.

Some important continuous distributions

- Continuous **uniform** distribution on interval $[a, b]$. The PDF is given by

$$p_X(x) = \frac{1}{b-a} \quad (x \in [a, b]).$$

  We write $X \sim \mathcal{U}[a, b]$.

- **Gaussian** distribution. With a location (mean) μ and scale (standard deviation) σ, the PDF is given by

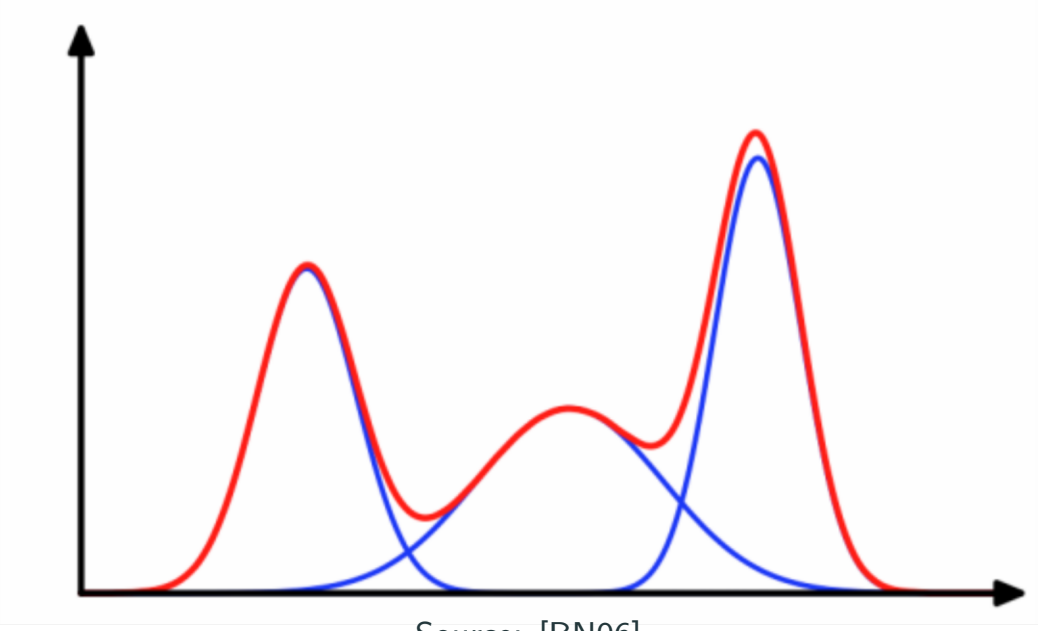
$$p_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (x \in \mathbb{R}).$$

  We write $X \sim \mathcal{N}(\mu, \sigma^2)$.

- **Gaussian mixture** distribution with K components. With K locations μ_k, scales σ_k and weights w_k, the PDF is given by:

$$p_X(x) = \sum_{k=1}^{K} w_k \mathcal{N}(\mu_k, \sigma_k^2),$$

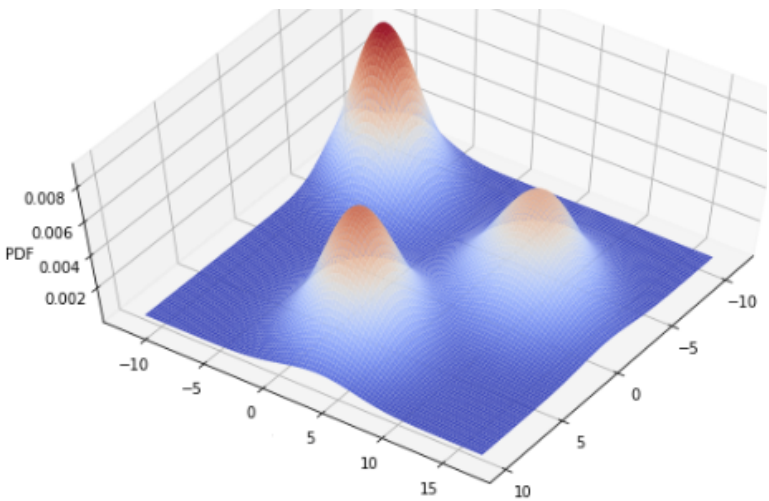
where $\sum_{k=1}^{K} w_k = 1$.

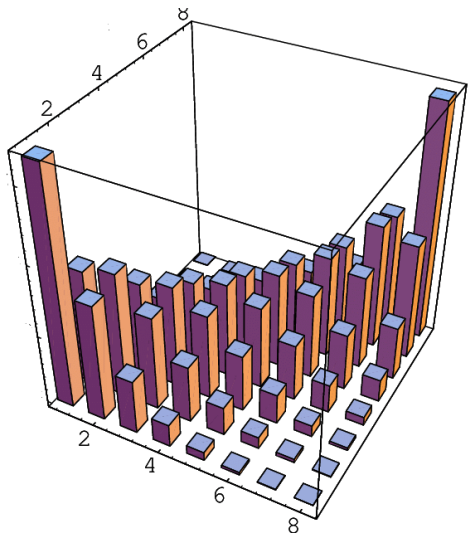


Source: [BN06]

More than one random variable?

- **Multivariate** random variables or **random vectors** are ubiquitous in modern data analysis.
- The uncertainty in the random vector is characterized by a **joint** PDF or PMF.





Joint distributions

- $p(x)$
- $p(x_1, x_2)$
- $p(x_1, x_2, x_3)$
- ...
- $p(x_1, \ldots, x_D)$
- We often just write $p(\boldsymbol{x})$ when the dimensionality is clear from context.

Random vectors

An image from a dataset can be represented by a **high-dimensional** vector.

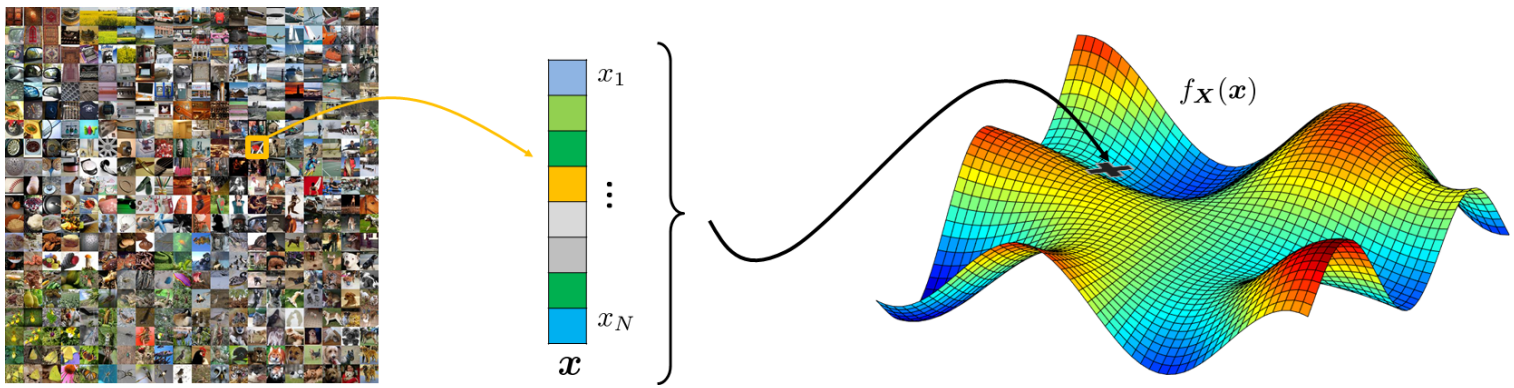


Image source: Introduction to Probability for Data Science, Stanley H. Chan.

Random vectors

Random vector:

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \dots \\ X_D \end{pmatrix} \text{ and } x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_D \end{pmatrix}$$

Joint PDF:

$$p_X(x) = p_{X_1, X_2, \dots, X_D}(x_1, x_2, \dots, x_D)$$

Probability:

$$\mathbb{P}(X \in A) = \int_A p_X(x) dx$$

Mean vector and covariance matrix

Let $X = (X_1, X_2, \ldots, X_D)^T$ be a random vector. The **expectation** is

$$\boldsymbol{\mu} = \mathbb{E}[X] = \begin{pmatrix} \mathbb{E}[X_1] \\ \mathbb{E}[X_2] \\ \ldots \\ \mathbb{E}[X_D] \end{pmatrix}.$$

The **covariance** matrix is

$$\boldsymbol{\Sigma} = \begin{pmatrix} \mathrm{Var}(X_1) & \mathrm{Cov}(X_1, X_2) & \ldots & \mathrm{Cov}(X_1, X_D) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Var}(X_2) & \ldots & \mathrm{Cov}(X_2, X_D) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(X_D, X_1) & \mathrm{Cov}(X_D, X_2) & \ldots & \mathrm{Var}(X_D) \end{pmatrix},$$

which can be written in a more compact way as

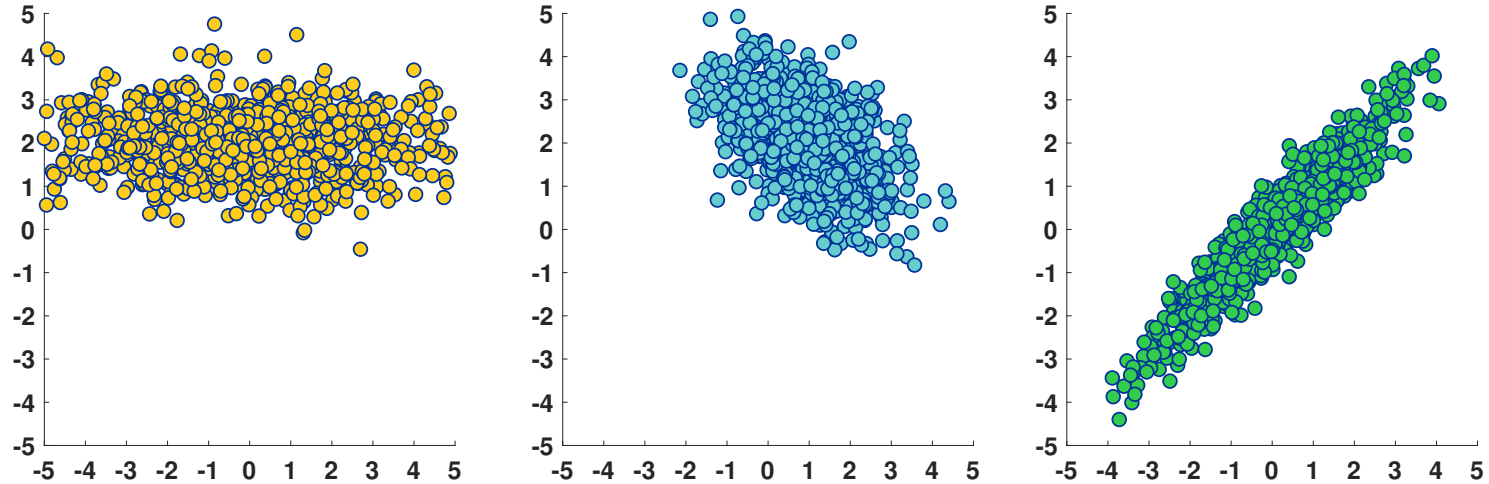
$$\boldsymbol{\Sigma} = \mathbb{E}[(X - \boldsymbol{\mu})(X - \boldsymbol{\mu})^T].$$

Multivariate Gaussian

A D-dimensional **joint Gaussian** has a PDF:

$$f_X(x) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} \exp\left\{ -\frac{1}{2}(x-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}) \right\}$$

Examples with $D = 2$:



$$(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 5 & 0 \\ 0 & 0.5 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 1.9 \\ 1.9 & 2 \end{bmatrix}$$

Deep Generative Modelling

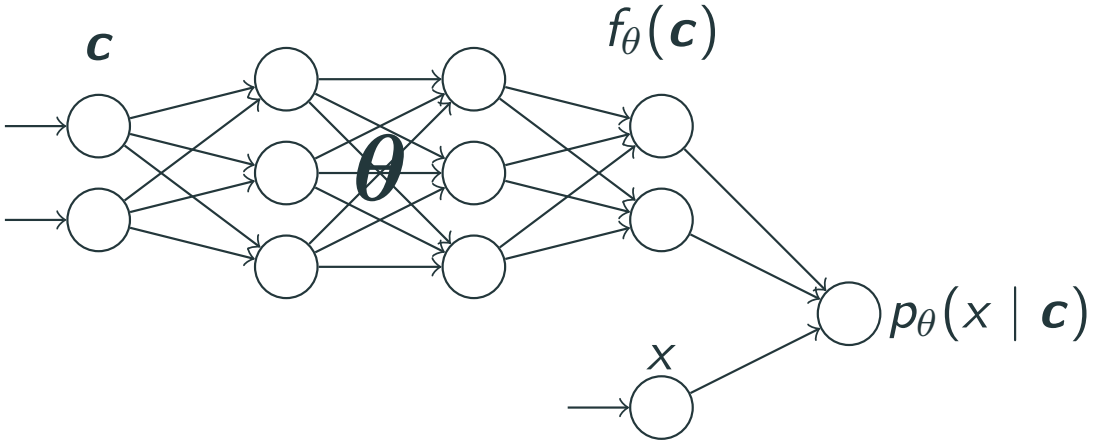
Given a dataset

$$\mathcal{D} = \left\{ (\boldsymbol{x}^{(1)}, \boldsymbol{c}^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, \boldsymbol{c}^{(n)}) \right\},$$

composed of n i.i.d. pairs sampled from $p(\boldsymbol{x}, \boldsymbol{c})$, we want to select parameters θ such that the distribution $p_\theta(\boldsymbol{x}|\boldsymbol{c})$ is a good approximation for $p(\boldsymbol{x}\ |\boldsymbol{c})$.

Neural networks for a categorical distribution ($D = 1$)

We can use neural networks to learn a probability distribution $p(x|\boldsymbol{c})$.



Categorical distribution

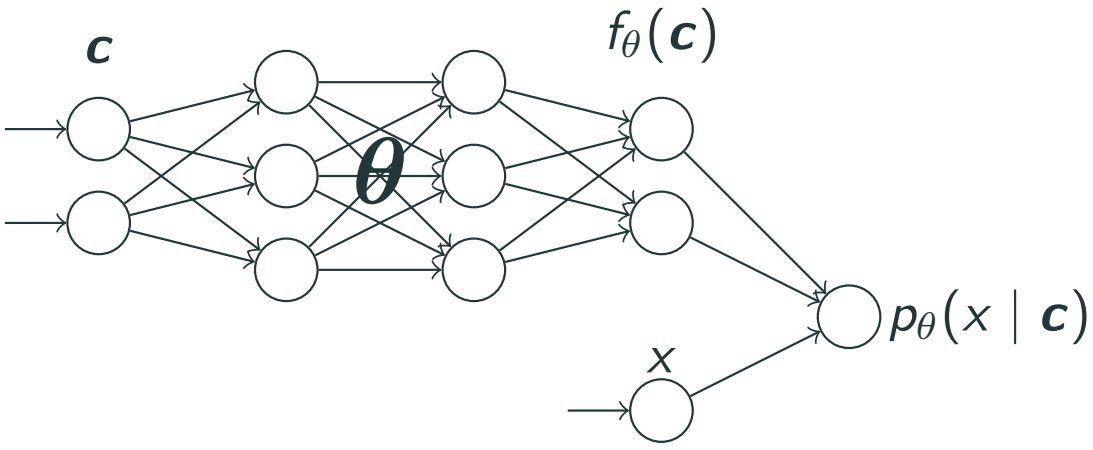
Given $\ell = f_\theta(\boldsymbol{c}) \in \mathbb{R}^K$ (we omit c and θ for ease of notation), we can compute a probability vector $\boldsymbol{p}$ using the *softmax* function:

$$\boldsymbol{p}_i = \frac{e^{\ell_i}}{\sum_{j=1}^{K} e^{\ell_j}} \quad \text{for } i = 1, \ldots, K,$$

and $p_\theta(x \mid \boldsymbol{c}) = \boldsymbol{p}_x$ where $\boldsymbol{p}_i \geq 0$ and $\sum_{i=1}^{K} \boldsymbol{p}_i = 1$.

Neural networks for a normal distribution ($D = 1$)

We can use neural networks to learn a probability distribution $p(x|\boldsymbol{c})$.



Normal distribution

Let $(\mu(\boldsymbol{c}), \rho(\boldsymbol{c})) = f_\theta(\boldsymbol{c})$ where $\mu(\boldsymbol{c}) \in \mathbb{R}$ and $\rho(\boldsymbol{c}) \in \mathbb{R}$. We can compute a strictly positive standard deviation $\sigma(\boldsymbol{c})$ using the *softplus* function:

$$\sigma(\boldsymbol{c}) = \log(1 + e^{\rho(\boldsymbol{c})}) > 0.$$

Then, we can compute $p_\theta(x \mid \boldsymbol{c}) = \mathcal{N}(x; \mu(\boldsymbol{c}), \sigma(\boldsymbol{c})^2)$.

Autoregressive models

Directly specifying a distribution for a random variable $\boldsymbol{x} \in \mathbb{R}^D$ in high dimension ($D >> 1$) is not easy.

The approach behind **autoregressive models** is to use the **product rule**:

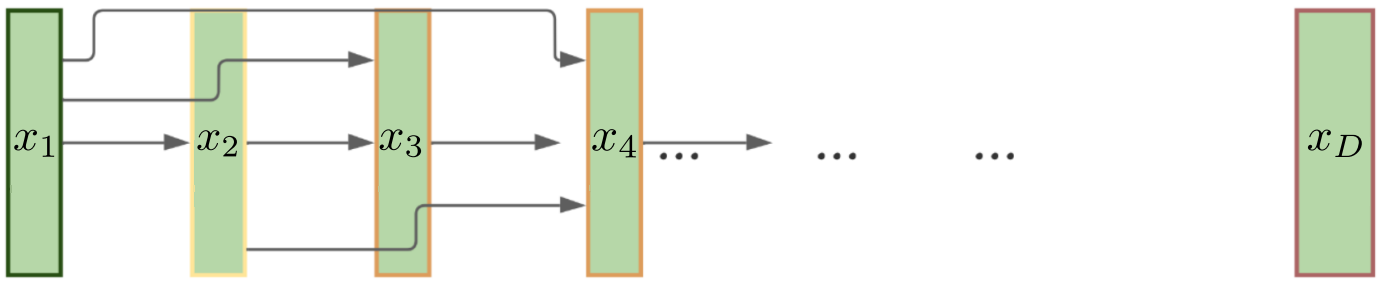
$$\begin{aligned} p(\boldsymbol{x}) &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2, x_1)\ldots \\ &= \prod_{d=1}^{D} p(x_d \mid x_1, \ldots, x_{d-1}) \end{aligned}$$

Autoregressive neural networks

For a given dimension d, we model the conditional density using a **neural network** $p_\theta(x_d \mid x_1, \ldots, x_{d-1})$.

$p_\theta(x_d \mid x_1, \ldots, x_{d-1})$ could be, for example, a categorical distribution or a normal distribution.

By doing so, we reduce the multidimensional estimation problem to a **unidimensional** one, which does not suffer from the curse of dimensionality.



Adapted from [Mur23]

Naive density evaluation and sampling

Given an auto-regressive model with parameters θ, how to evaluate its density $p_\theta(\boldsymbol{x})$ and sample $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$?

Density evaluation:

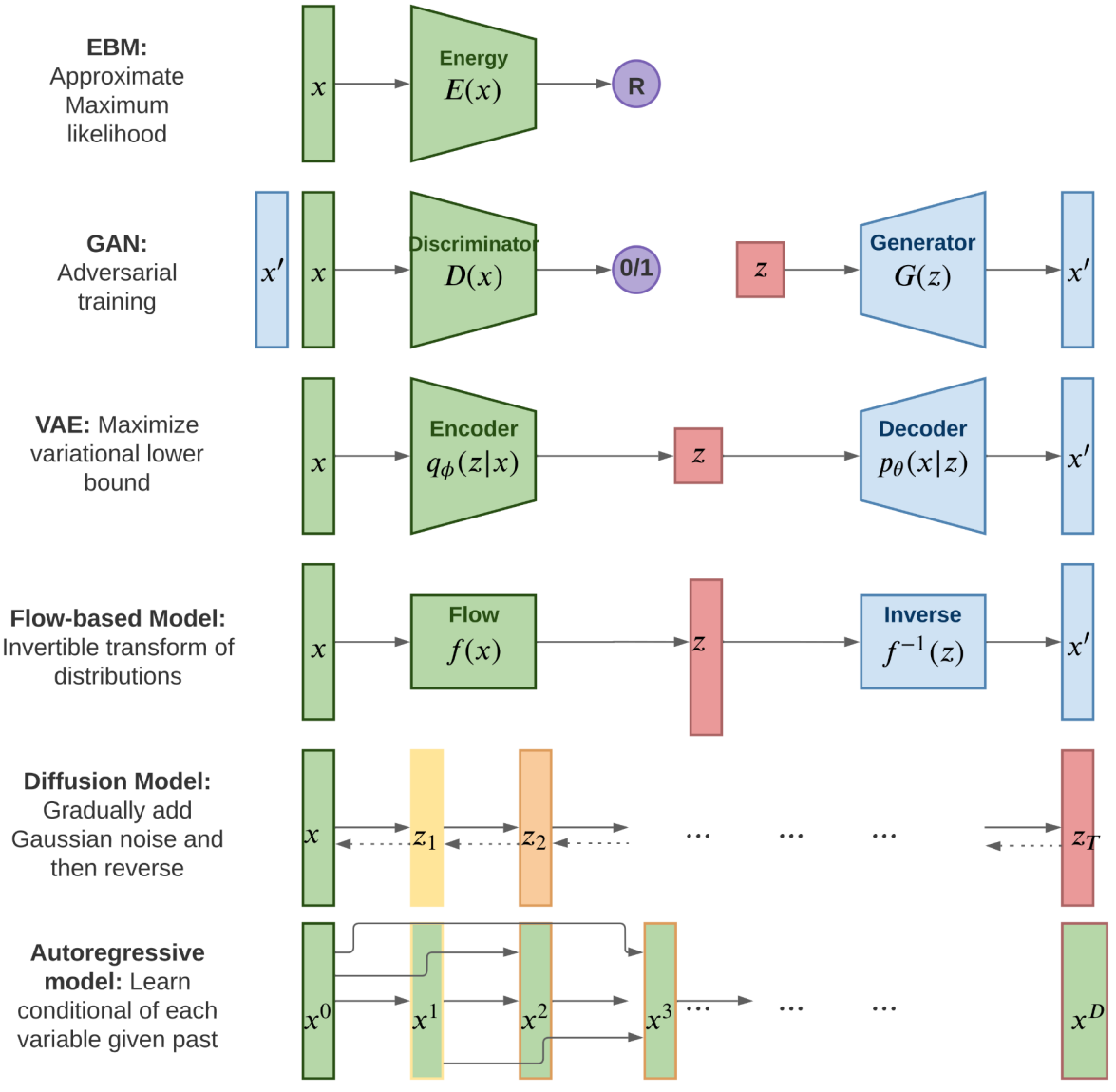
- Evaluate $p_\theta(x_1)$
- Evaluate $p_\theta(x_2 \mid x_1)$
- ...
- Evaluate $p_\theta(x_D \mid x_1, \ldots, x_{D-1})$

Sampling:

- Sample $x_1 \sim p_\theta(x_1)$
- Sample $x_2 \sim p_\theta(x_2 \mid x_1)$
- ...
- Sample $x_D \sim p_\theta(x_D \mid x_1, \ldots, x_{D-1})$

Both operations require D neural network forward passes.

Overview of deep generative models



Source: [Mur23]

Maximum Likelihood Estimation

Setting

A dataset of n observations from $p(\boldsymbol{x})$ is given:

$$\mathcal{D} = \left\{ \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)} \right\} \subseteq \mathcal{X}. \tag{1}$$

The standard assumption is that the observations are **independent and identically distributed (i.i.d.)**.

We want to select parameters θ such that the distribution $p_\theta(\boldsymbol{x})$ is a good approximation to $p(\boldsymbol{x})$.

Maximum likelihood estimation (MLE)

The maximum likelihood estimator is given by:

$$\arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\theta} p_\theta(\mathcal{D}) \qquad \text{(Definition of likelihood)}$$

Maximum likelihood estimation (MLE)

The maximum likelihood estimator is given by:

$$\begin{aligned}
\arg\max_{\theta} \mathcal{L}(\theta) &= \arg\max_{\theta} p_\theta(\mathcal{D}) && \text{(Definition of likelihood)} \\
&= \arg\max_{\theta} \prod_{i=1}^{n} p_\theta(\boldsymbol{x}^{(i)}) && \text{(Independence)}
\end{aligned}$$

Maximum likelihood estimation (MLE)

The maximum likelihood estimator is given by:

$$\begin{aligned}
\arg\max_\theta \mathcal{L}(\theta) &= \arg\max_\theta p_\theta(\mathcal{D}) && \text{(Definition of likelihood)} \\
&= \arg\max_\theta \prod_{i=1}^{n} p_\theta(\boldsymbol{x}^{(i)}) && \text{(Independence)} \\
&= \arg\max_\theta \log \prod_{i=1}^{n} p_\theta(\boldsymbol{x}^{(i)}) && \text{(log is increasing)}
\end{aligned}$$

Maximum likelihood estimation (MLE)

The maximum likelihood estimator is given by:

$$\begin{aligned}
\arg\max_\theta \mathcal{L}(\theta) &= \arg\max_\theta p_\theta(\mathcal{D}) && \text{(Definition of likelihood)} \\
&= \arg\max_\theta \prod_{i=1}^{n} p_\theta(\boldsymbol{x}^{(i)}) && \text{(Independence)} \\
&= \arg\max_\theta \log \prod_{i=1}^{n} p_\theta(\boldsymbol{x}^{(i)}) && \text{(log is increasing)} \\
&= \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(\boldsymbol{x}^{(i)}) && \text{(log of product)}
\end{aligned}$$

Maximum likelihood estimation (MLE)

The maximum likelihood estimator is given by:

$$
\begin{aligned}
\arg\max_\theta \mathcal{L}(\theta) &= \arg\max_\theta p_\theta(\mathcal{D}) && \text{(Definition of likelihood)} \\
&= \arg\max_\theta \prod_{i=1}^{n} p_\theta(\boldsymbol{x}^{(i)}) && \text{(Independence)} \\
&= \arg\max_\theta \log \prod_{i=1}^{n} p_\theta(\boldsymbol{x}^{(i)}) && \text{(log is increasing)} \\
&= \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(\boldsymbol{x}^{(i)}) && \text{(log of product)} \\
&= \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} -\log p_\theta(\boldsymbol{x}^{(i)}) && (x \mapsto -x/n \text{ is decreasing})
\end{aligned}
$$

MLE as KL divergence minimization

By the weak law of large numbers:

$$\frac{1}{n}\sum_{i=1}^{n} -\log p_\theta(\boldsymbol{x}^{(i)}) \xrightarrow{p} \underbrace{\mathbb{E}_{p(\boldsymbol{x})}[-\log p_\theta(\boldsymbol{x})]}_{\text{Expected NLL}} \quad \text{when } n \to \infty.$$

We can think of NLL minimization as trying to minimize the **KL divergence** between the true distribution $p(\boldsymbol{x})$ and the estimated distribution $p_\theta(\boldsymbol{x})$:

$$\begin{aligned}
\arg\min_\theta \mathbb{E}_{p(\boldsymbol{x})}[-\log p_\theta(\boldsymbol{x})] &= \arg\min_\theta \mathbb{E}_{p(\boldsymbol{x})}[\log p(\boldsymbol{x}) - \log p_\theta(\boldsymbol{x})] \\
&= \arg\min_\theta \mathbb{E}_{p(\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x})}{p_\theta(\boldsymbol{x})}\right] \\
&= \arg\min_\theta D_{\mathrm{KL}}(p(\boldsymbol{x}) \parallel p_\theta(\boldsymbol{x}))).
\end{aligned}$$

More data ($n \to \infty$) leads to a better approximation.

Properties of KL divergence

Basic properties of the **Kullback-Leibler (KL) divergence**:

$$D_{\mathrm{KL}}(p \parallel q) = \mathbb{E}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right] = \int p(x) \log \frac{p(x)}{q(x)} dx.$$

Note that $D_{\mathrm{KL}}(p \parallel q) \geq 0$ for any p, q:

$$\begin{aligned} D_{\mathrm{KL}}(p \parallel q) &= -\mathbb{E}_{p(x)}\left[\log \frac{q(x)}{p(x)}\right] \\ &\geq -\log \mathbb{E}_{p(x)}\left[\frac{q(x)}{p(x)}\right] && \text{(Jensen's inequality)} \\ &= -\log \int p(x) \frac{q(x)}{p(x)} dx \\ &= -\log 1 && \text{(integral of PDF)} \\ &= 0. \end{aligned}$$

If $p = q$, $D_{\mathrm{KL}}(p \parallel q) = 0$.

Conditional MLE

We have a dataset $\mathcal{D} = \left\{ (\boldsymbol{x}^{(1)}, \boldsymbol{c}^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, \boldsymbol{c}^{(n)}) \right\}$ where $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ are assumed **i.i.d. conditional on** $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(n)}$.

We compute the MLE as follows:

$$\begin{aligned} \arg\max_{\theta} \mathcal{L}(\theta) &= \arg\max_{\theta} p_\theta(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)} \mid \boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(n)}) \\ &= \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} - \log p_\theta(\boldsymbol{x}^{(i)} \mid \boldsymbol{c}^{(i)}). \end{aligned}$$

We successively update the parameters using gradient descent:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} - \log p_\theta(x^{(i)} \mid \boldsymbol{c}^{(i)}),$$

where $\eta > 0$ is the learning rate.