Generative modelling Summer School (GeMSS)

26th-30th June, Copenhagen

Organised by J. Frellsen, Mattei P. A. and Tomczak J. M.

What are we trying to solve?

Probability distributions are the right object to model random phenomena, but **they are not very convenient, because defining functions whose inputs are subsets is not easy !**

Given an **unknown distribution** $\mathbb{P}_{data}$ **with density** p_{data}

We try to estimate **parameters** $\hat{\theta} \in \Theta$ such that $p_{\hat{\theta}} \approx p_{data}$

How do we do that?

2 ways:

Bayesian inference

Maximum Likelihood estimation

How do we do that?

2 ways:

Bayesian inference

Maximum Likelihood estimation

Bayesian inference

Idea: Learning a prior that gives higher probability to the models you find best before seeing the data

Posterior density $p(\theta \mid x) = \dfrac{p(x \mid \theta) p_\Theta(\theta)}{p(x)}$ **Prior**

Conceptually very simple but difficult to perform in practice

How do we do that?

2 ways:

Bayesian inference

Maximum Likelihood estimation

Maximum likelihood estimation

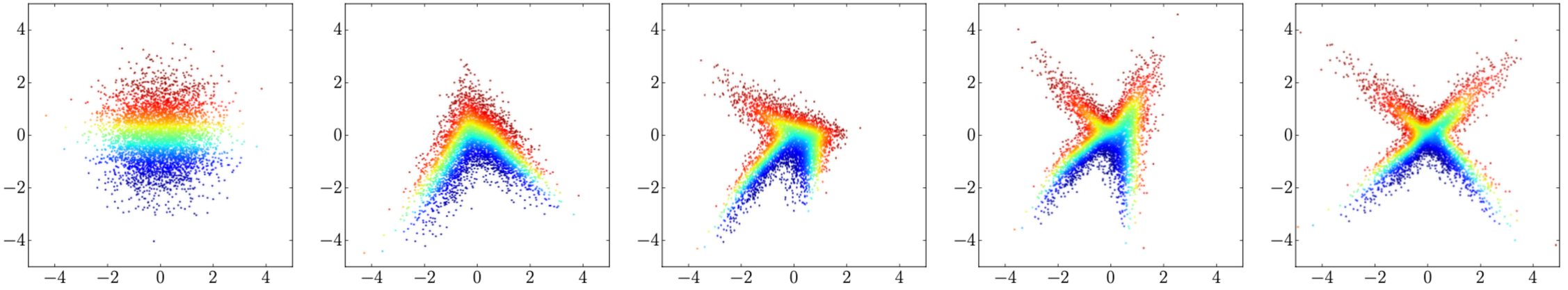
Idea: Minimising distance d over models and find the model closest to the truth.

$$\hat{\theta} \in \arg\min_{\theta} d(p_{data}, p_{\theta})$$

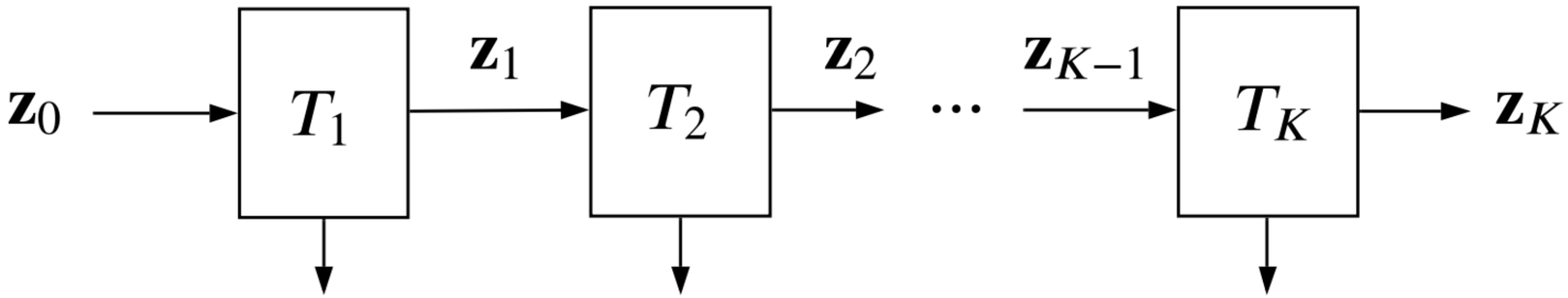
Mostly used approach in all current DL models:
VAEs, GANs, ARMs, diffusion, ...

Normalizing flows

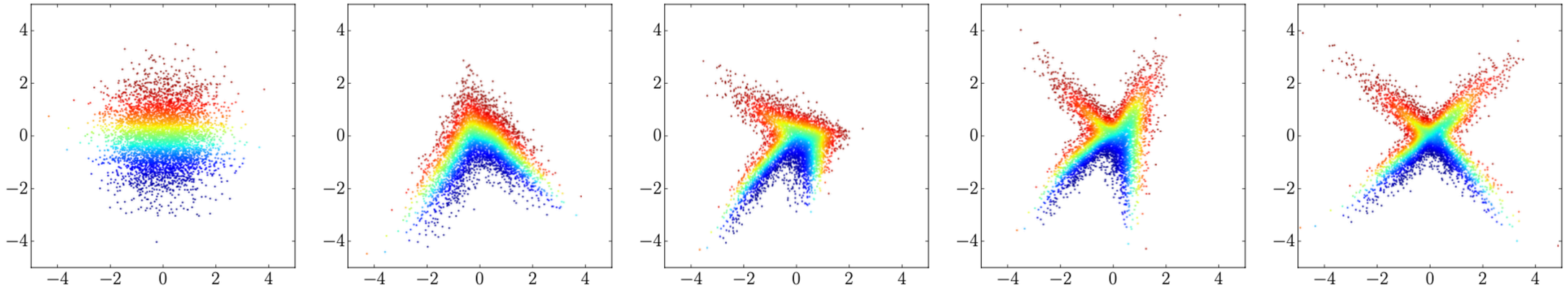
- "Flow": Refers to the sequence of transformation $T = T_K \circ \cdots \circ T_1$ where a sample $\mathbf{u}$ *flows* from the base to the output $\mathbf{x}$
- "Normalising": Refers to the reverse flow where a datum $\mathbf{x}$ *flows* back to a normal distributed base, i.e. data is normalised



Normalizing flows



$$\log|\det J_{T_1}(\mathbf{z}_0)| + \log|\det J_{T_2}(\mathbf{z}_1)| + \cdots + \log|\det J_{T_K}(\mathbf{z}_{K-1})| = \log|\det J_T(\mathbf{z}_0)|$$



Normalizing flows

What can we use them for?

- **Density estimation**
  - With **exact likelihood**
  - Can **approximate any distribution**
  - But, layers are simple, so many are required (c.f. Glow)
- **Variational inference**
  - A flow can be a very **flexible approximate posterior**
- **Flexible priors in VAEs**
- **Likelihood free inference**
  - Can be used to **approximate the likelihood or posterior** in ABC
- ... and more (see section 6 in Papamakarios et al.[1])

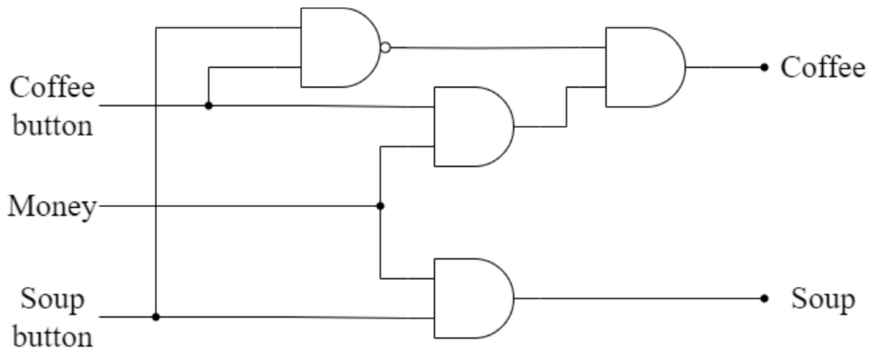
[1]Papamakarios G, Nalisnick E, Rezende DJ, Mohamed S, Lakshminarayanan B. Normalizing Flows for Probabilistic Modeling and Inference. JMLR, 2021.

Probabilistic circuits
(Sum-Product Networks)

You probably know **Logic circuits:**

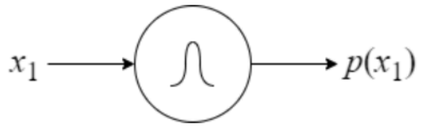
- **Example**: Vending machine

INPUTS			OUTPUTS	
Coffee Button	Money	Soup Button	Coffee	Soup
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	0	0
1	1	0	1	0
1	1	1	0	1

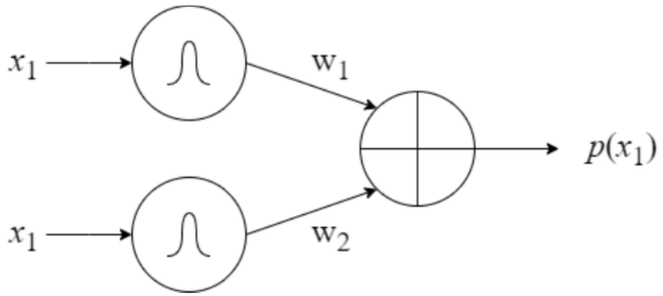


Probabilistic circuits
(Sum-Product Networks)

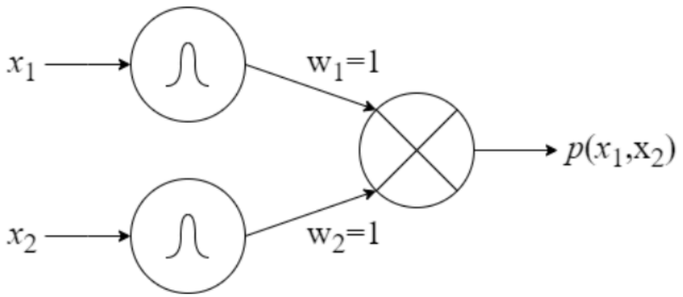
- Similarly to logic units, we could formulate probabilistic units, however, with different semantics:
  - **Inputs** are random variables (their values) or probabilities (also **outputs**).
  - **Nodes**: either a distribution, a summation or a product.
  - **Edges** define directions of probability flows.



"Buffer" unit

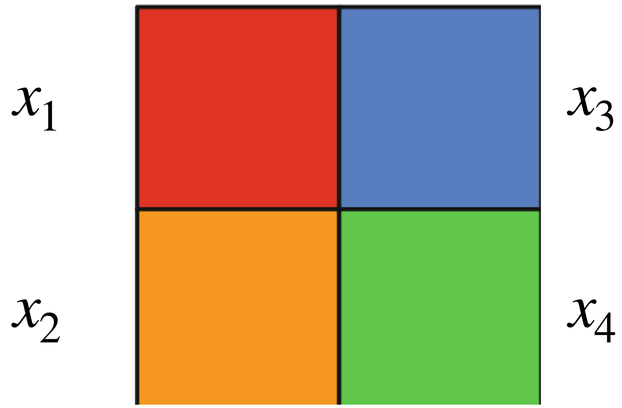


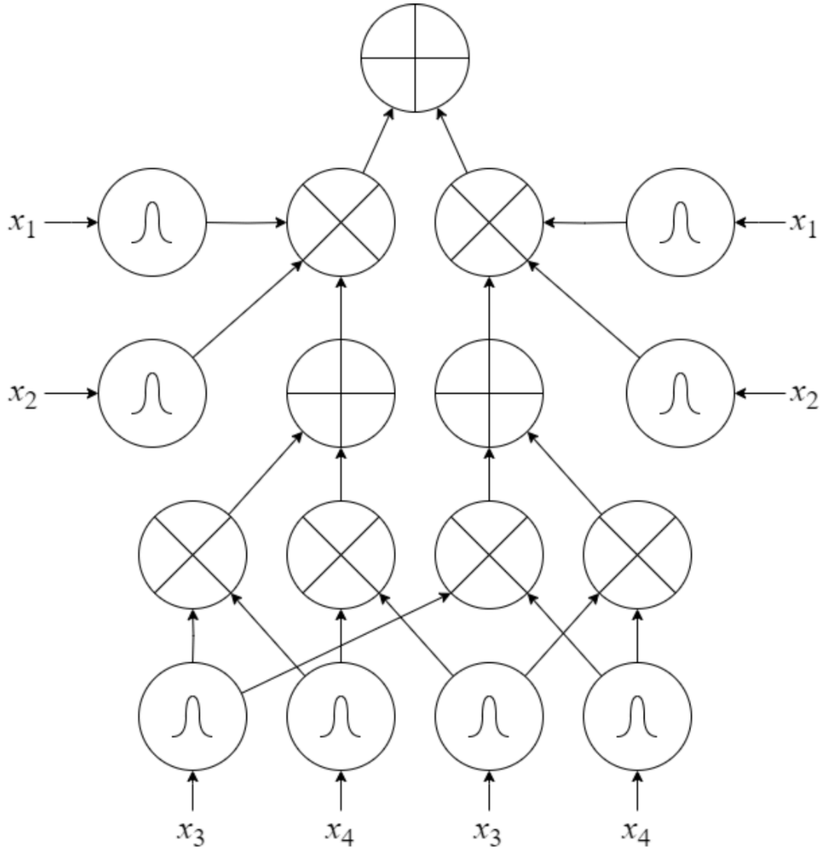
"OR" unit



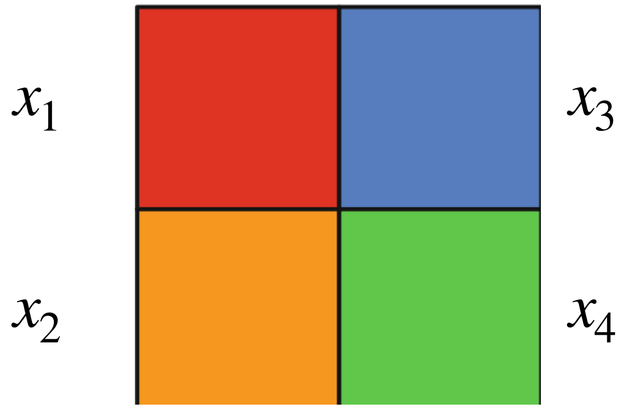
"AND" unit

Probabilistic circuits
(Sum-Product Networks)

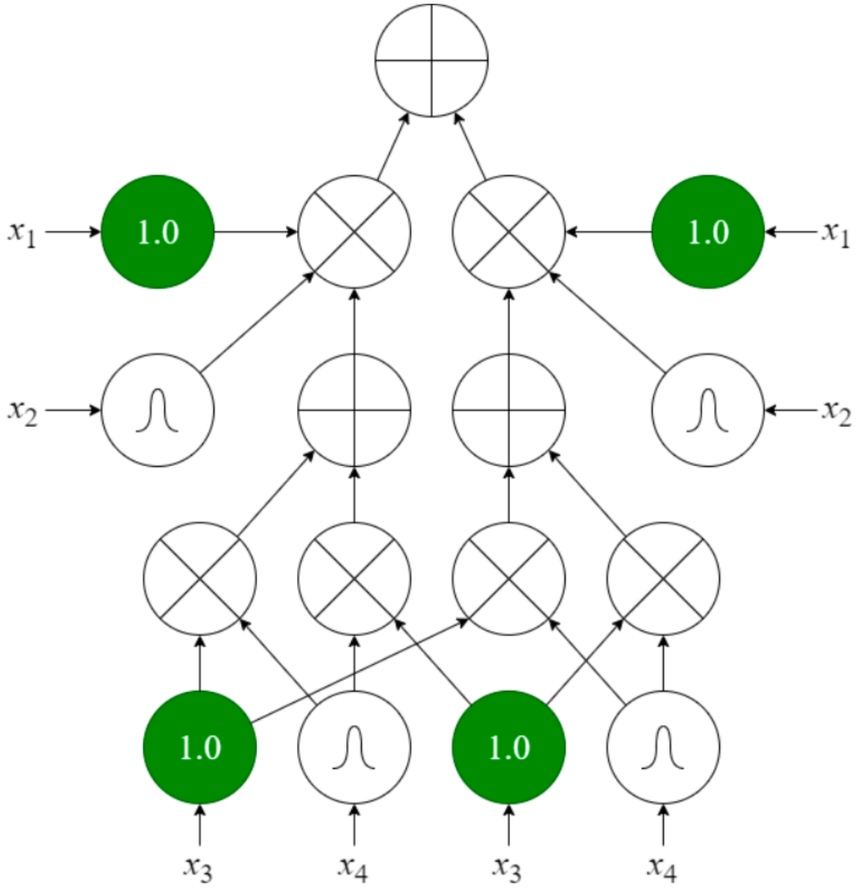




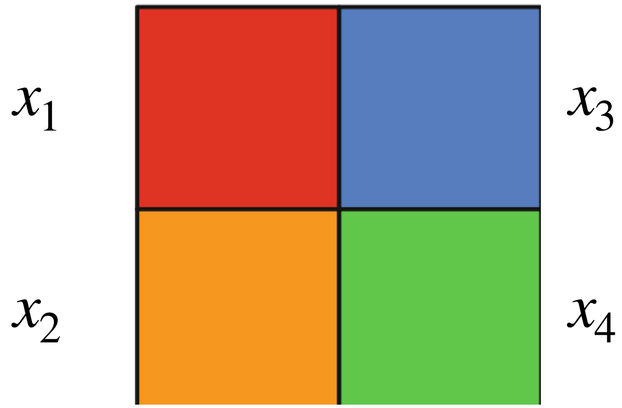
Probabilistic circuits
(Sum-Product Networks)



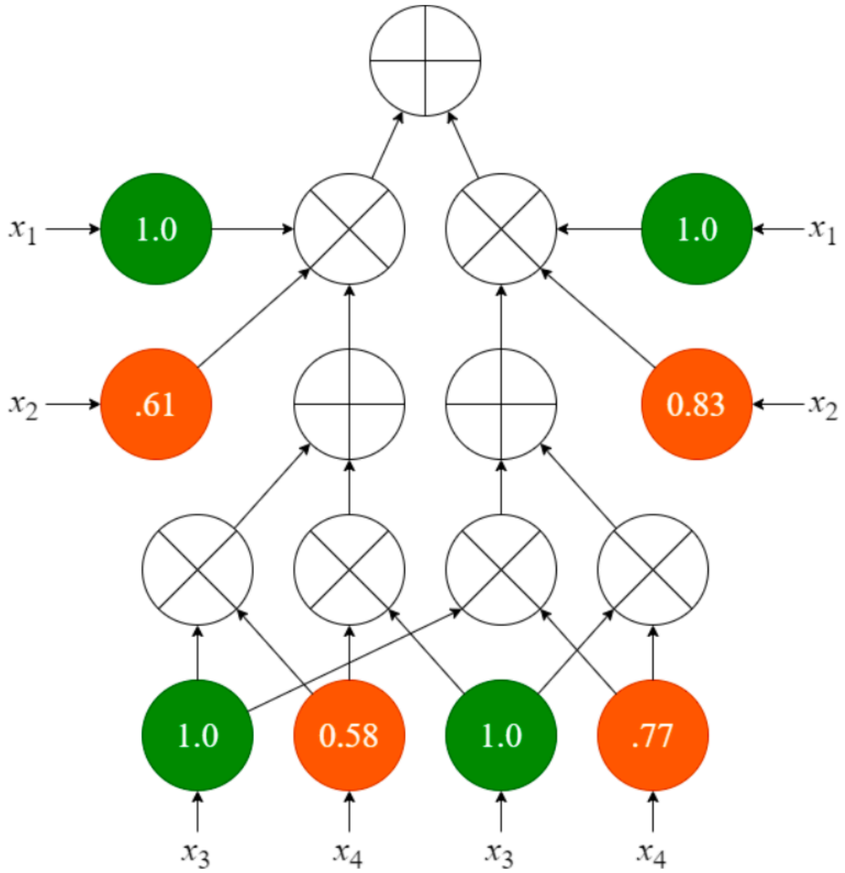
Compute evidence $p(x_2, x_4)$



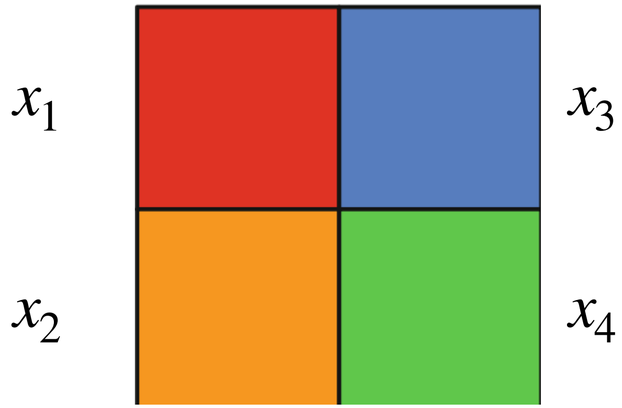
Probabilistic circuits
(Sum-Product Networks)



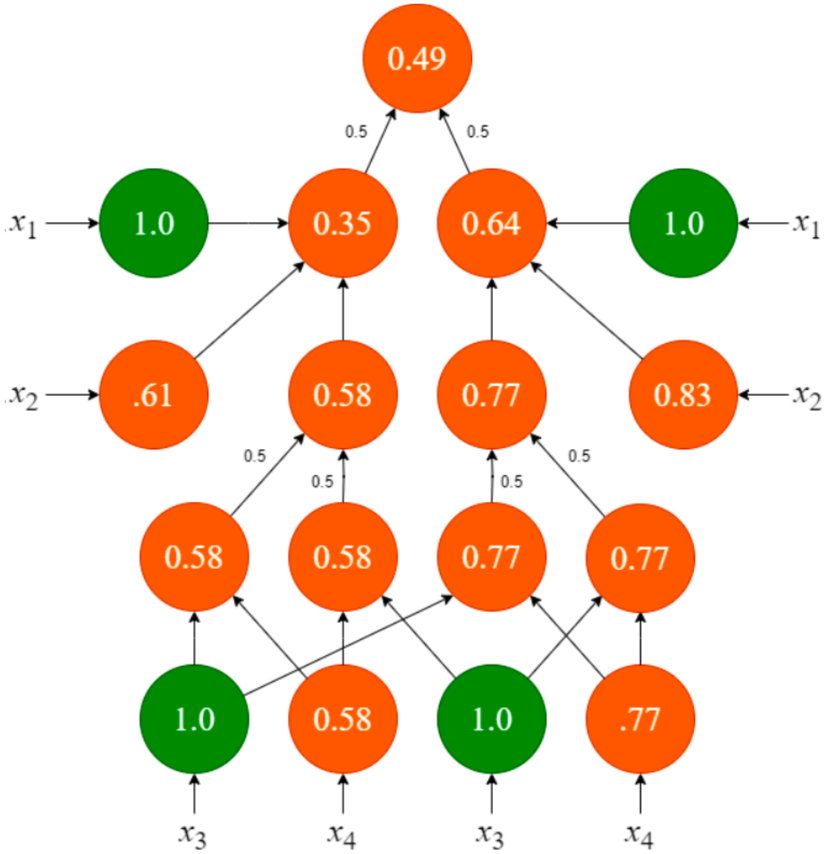
Compute evidence $p(x_2, x_4)$



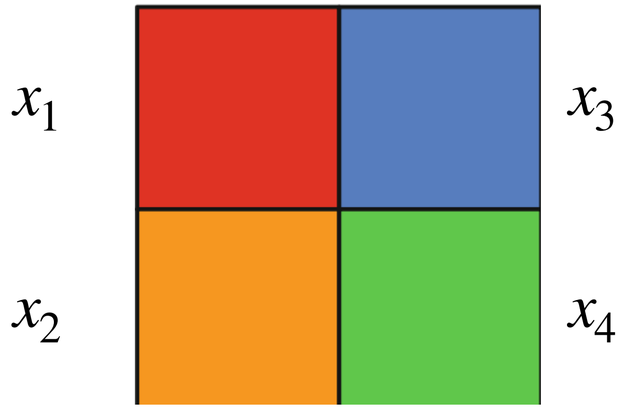
Probabilistic circuits
(Sum-Product Networks)



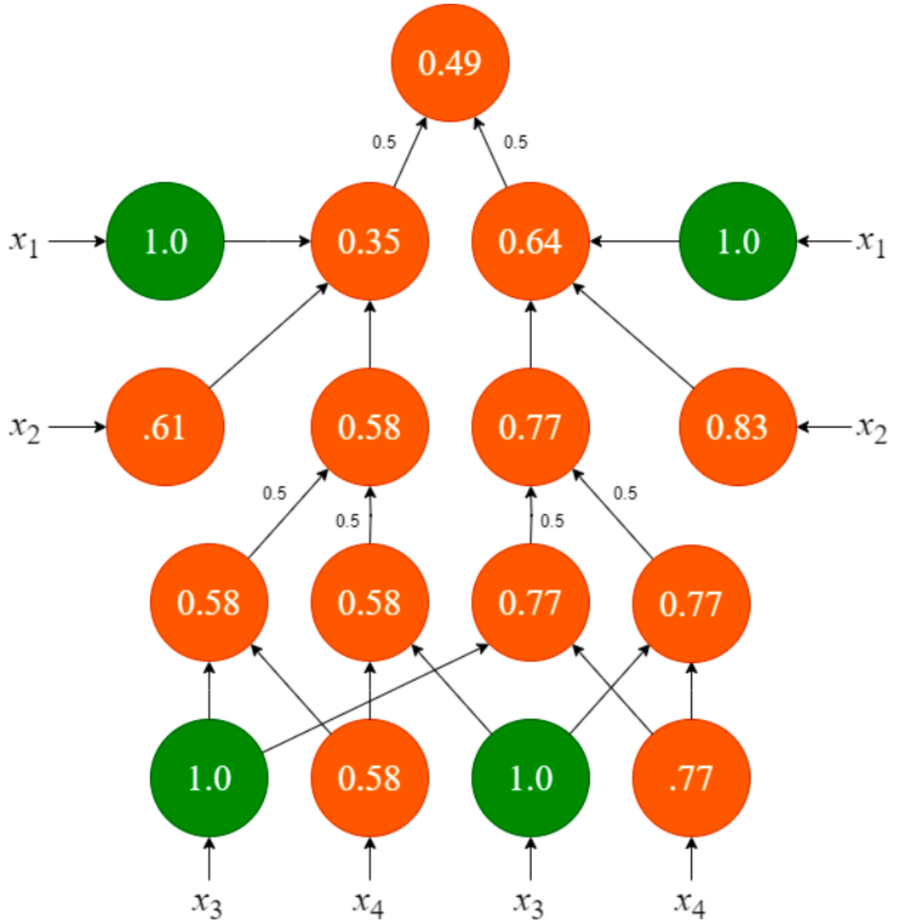
Compute evidence $p(x_2, x_4)$



Probabilistic circuits
(Sum-Product Networks)



Compute evidence $p(x_2, x_4)$

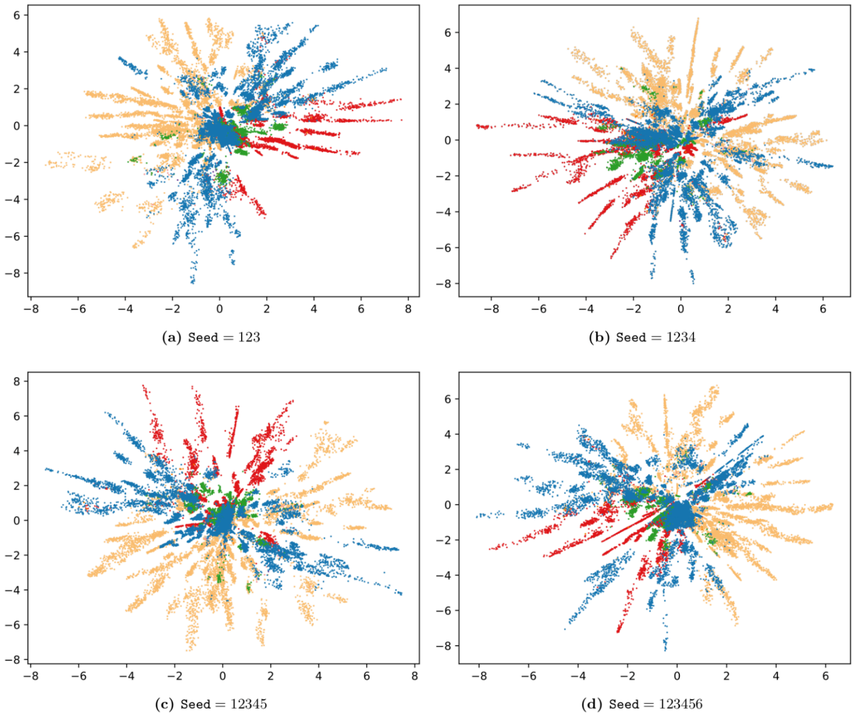


Identifiability and invariance using latent geometry

Invited lecturer: Søren Hauberg

- Pb: Finding a solution to the identifiability problem when learning latent representations

When fitting a generative model to data, we often find that different training runs give different latent representations.



(a) Seed = 123

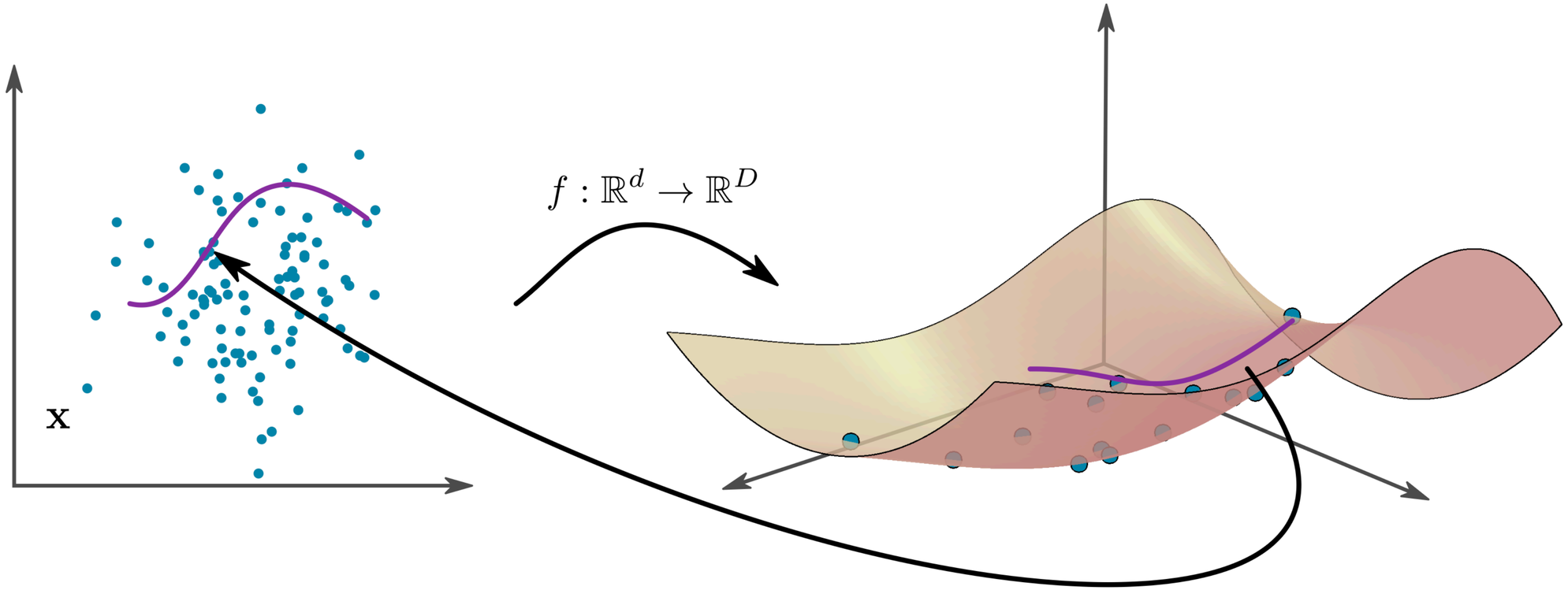
(b) Seed = 1234

(c) Seed = 12345

(d) Seed = 123456

Identifiability and invariance using latent geometry

Invited lecturer: Søren Hauberg

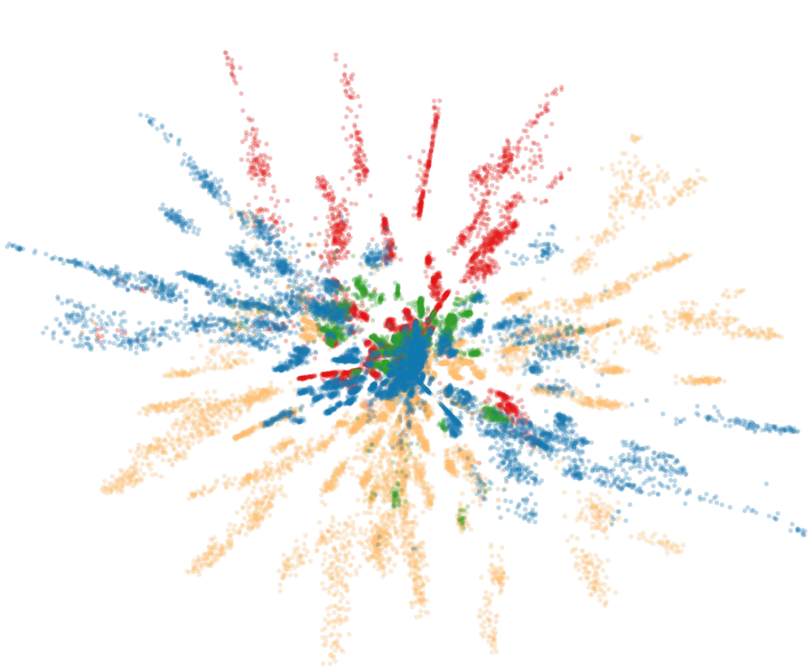


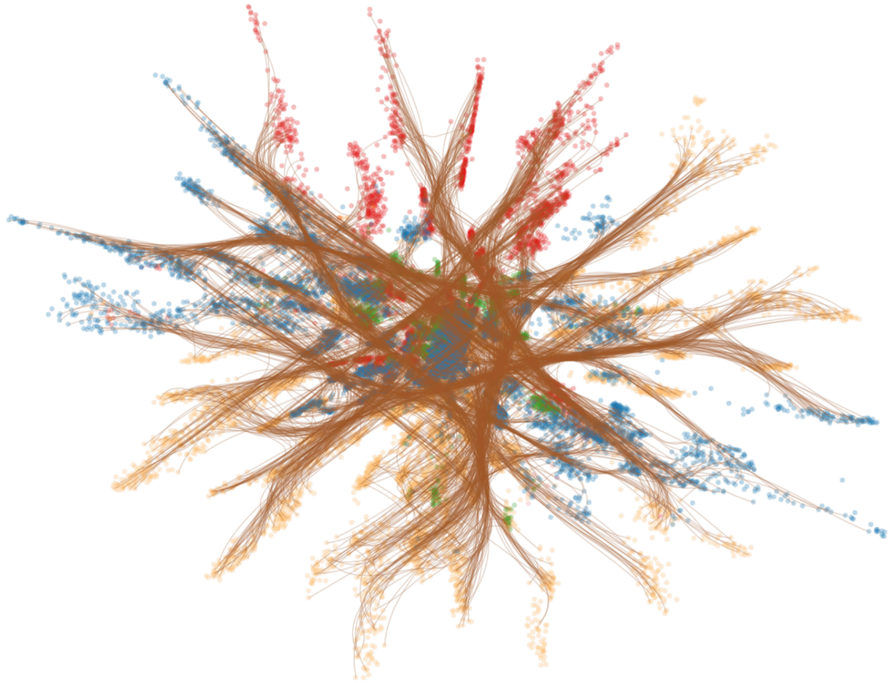
Define length of *latent curve* to be the
length of the *decoded curve*.
From that definition, you just go where the math takes you...

Identifiability and invariance using latent geometry

Invited lecturer: Søren Hauberg

Each point is a member of the protein family beta-lactamase represented in the latent space of a VAE



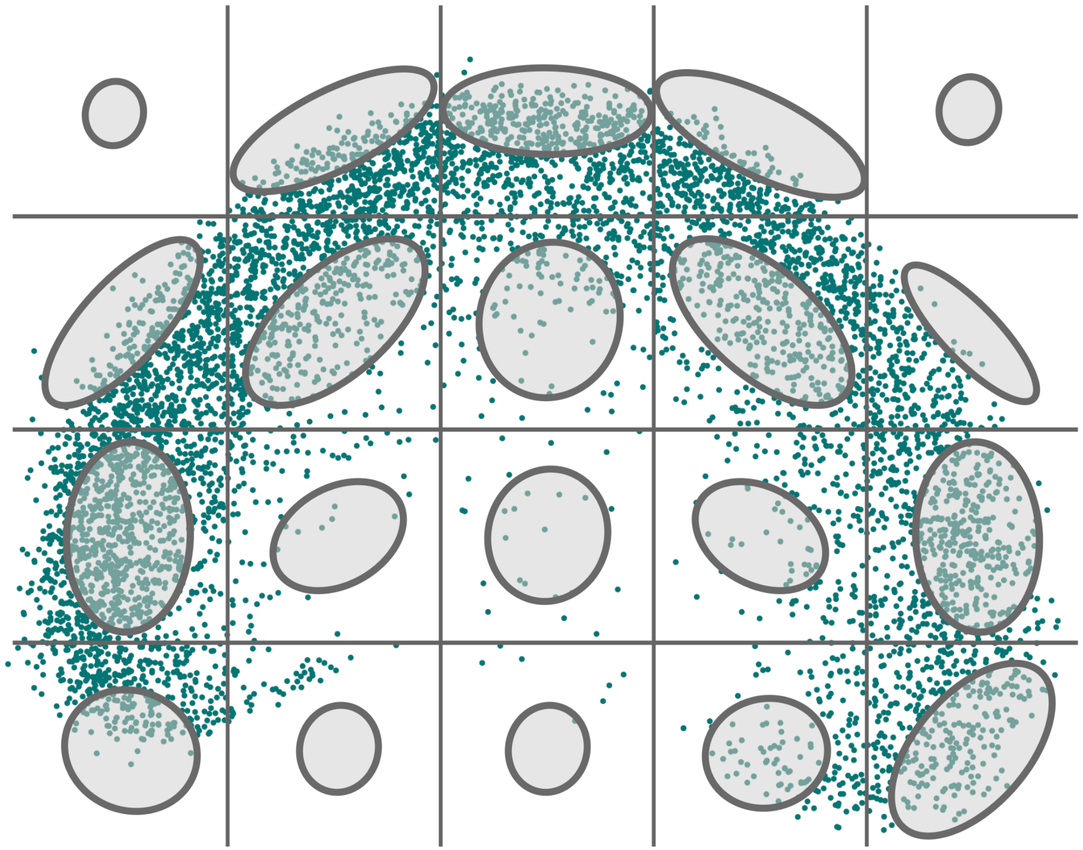


The change of definition of distance is allowed by using a different metric in the latent space: **the Riemann metrics**

Identifiability and invariance using latent geometry

Invited lecturer: Søren Hauberg

Since the latent representation now has a local norm, local unit circles are now different across the representation:



This is the biggest conceptual difference to the standard Euclidean interpretation.

Other interesting topics

- Energy-based model (by Will Grathwohl, Deepmind)
- Causal discovery and inference (by Cheng Zhang, Microsoft)

  → *“Creating good benchmarks is very challenging and most benchmark are not adapted”*
- Latent diffusion (by Robin Rombach, Stability AI)

  → *“Next steps:*

  - 3D generation (e.g. DreamFusion),
  - video generation,
  - new representation ? (Spatially adaptative, abstract concepts, motion vectors, ...)
  - watermarking
- Probabilistic circuits (by Robert Peharz, TU Graz)