Validation of generative models, especially SBI

- fundamental problem: if $x \sim p(x)$, there is no single correct outcome
  - traditional fitting $\hat{x}_i = x_i^*$ does not work

⇒ must compare distributions, not individual outcomes

case (1) we have a GT sample $\{x_i^* \sim p^*(x)\}_{i=1}^N$ or $\{x_i^* \sim p^*(x|y)\}_{i=1}^N$ for fixed y

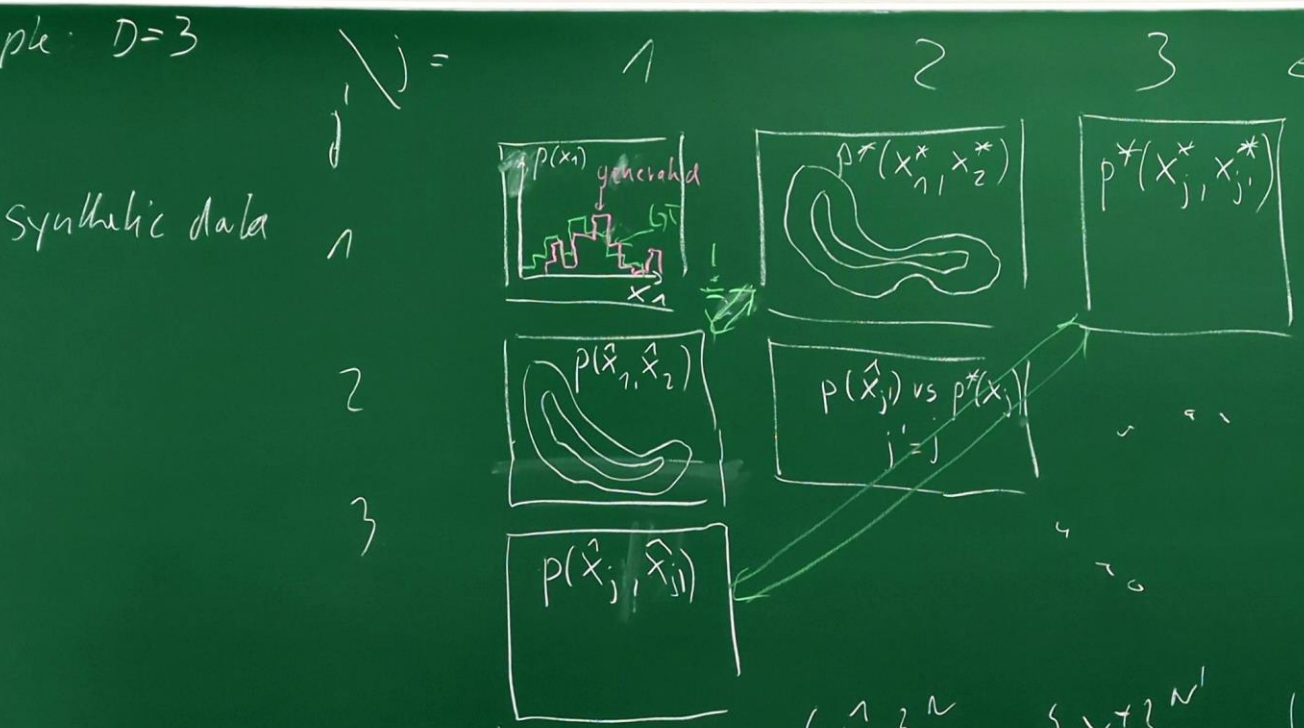
easily generated by simulation $\Sigma = \int p^*(y)\, p^*(x|y)\, dy$

[more difficult for posterior $p(y|x) \approx p^*(y|x)$

⇒ to generate $\{y_i^* \sim p^*(y|x)\}_{i=1}^N$ for fixed x^{obs}, we need a classical algorithm like MCMC or ABC for $y \sim p^*(y|x^{obs})$]

- compare covariances of $\{\hat{x}_i\}$ and $\{x_i^*\}$: quick check, complete if p is Gauss
  - means and
- plot marginal distributions in 1D and 2D for features $j, j' = 1, \ldots, D$

example: $D=3$



check visually that corresponding plots are similar

- various scores to compare the samples $\{\hat{x}_i\}_{i=1}^{N}$ $\{x^*_i\}_{i=1}^{N'}$ (usually, $N'=N$)
  - MMD (maximum mean discrepancy)

$$MMD^2 = \frac{1}{N'(N'-1)} \sum_{i=1}^{N'} \sum_{i' \neq i} k(x^*_i, x^*_{i'}) + \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{i' \neq i} k(\hat{x}_i, \hat{x}_{i'})$$

$$- \frac{2}{N \cdot N'} \sum_{i=1}^{N'} \sum_{i'=1}^{N} k(x^*_i, \hat{x}_{i'})$$

with kernel $k(x, x')$

- Frechet Inception Distance (FID) especially popular if X is an image

idea: - compare means and covariance matrices via Frechet distance (= Wasserstein 2 distance)

- if data are Gaussian with $\hat{\mu}, \hat{\Sigma}$ and μ^*, Σ^* for GT

Frechet distance can be computed analytically

$$FD = \|\hat{\mu} - \mu^*\|_2^2 + \mathrm{tr}\left[\hat{\Sigma} + \Sigma^* - 2(\hat{\Sigma}\Sigma^*)^{1/2}\right]$$

for images: Gaussian assumption is not fulfilled $\tilde{\hat{x}}_i = h(\hat{x}_i)$, $\tilde{x}_i^* = h(x_i^*)$

$\Rightarrow$ instead calculate FD in some feature space

fit $\hat{\mu}, \hat{\Sigma}$ and μ^*, Σ^* to $\tilde{\hat{x}}$ and $\tilde{x}^*$ features

definition of $h(x)$: - typically, use a pre-trained network

traditionally: Inception network for ImageNet

today: use some foundational model, e.g. CLIP (OpenAI)
(trained with self-supervision on 10^9 images) DINOv2 (Meta)

famous algorithm for self-supervised learning: SIMCLR7

- if no pre trained model available: use a random (ly initialized) network

- density & coverage (heuristic version of MMD²) [Naeem et al. 2020]

use nearest neighbor method, usually applied in a feature space to make Euclidean distance plausible

define $B_k(X_i)$: ball with center X_i and radius $\|X_i - X_{i,[k]}\|_2$

$\Rightarrow \mathbb{1}[X' \in B_k(X_i)]$ is a kernel!

$X_{i,[k]}$: k^{th} nearest neighbor of X_i

$$\text{density} = \frac{1}{kN'} \sum_{i'=1}^{N'} \sum_{i=1}^{N} \mathbb{1}[\hat{X}_i \in B_k(X^*_{i'})]$$

counts how many $\hat{X}_i$ each ball around $X^*_{i'}$ contains

$\Rightarrow$ high when the $\hat{X}_i$ are close to the X^*_i

$$\text{coverage} := \frac{1}{N'} \sum_{i=1}^{N'} \mathbb{1}[\exists\, i \text{ such that } \hat{X}_i \in B_k(X^*_{i'})]$$

counts how many of the balls around X^*_i contain a $\hat{X}_i$

[recommended hyperparameters: $N = N' = 10000$, $k = 5$]

asymptotic limits when $p(x) = p^*(x)$ (distributions are identical)

$\mathbb{E}[\text{density}] = 1$ $\qquad$ $\mathbb{E}[\text{coverage}] = 1 - 1/2^k$ $\qquad$ (if $N \to \infty$)

relation to MMD: $\text{density}(\hat{X}, \hat{X}) = \text{density}(X^*, X^*) = 1 \Rightarrow$ $\boxed{MMD^2 \overset{\sim}{=} 2 - \text{density}(\hat{X}, X^*) - \text{density}(X^*, \hat{X})}$

case (2) we do not have a GT sample $\{x_i^*\}_{i=1}^N$

$\Rightarrow$ must evaluate on the basis of $\{\hat{x}_i\}_{i=1}^N$ only, possible with a single instance $x^* \sim p^*(x)$

[e.g. when learning the posterior, we have $\{(y_i^* \sim p^*(y),\ x_i^* = \phi(y_i, m_i))\}_{i=1}^N$

then y_i^* is a single GT example for $\hat{y} \sim p(y \mid x_i^*)$]

- Check the diversity of the generated sample (without any GT)

  Vendi score [Friedman & Dieng 2023]

  kernel-based method like MMD) diversity = $\{\hat{x}_i\}_{i=1}^N$ are all different, ideally cover entire $p^*(x)$

  algorithm (1) calculate kernel (Gram) matrix G:

  $$G_{ii'} = \frac{1}{N} k(\hat{x}_i, \hat{x}_{i'})$$ (can also be done in a feature space $\psi(x)$)

  [(1a) missing in the paper: centralize G as in kernel PCA (see MLE)

  $$G \leftarrow C_N G C_N \qquad C_N = (\mathbb{I}_N - \frac{1}{N} \mathbb{1}\mathbb{1}^T)$$]

  (2) calculate EV of G: λ_1 ($\operatorname{tr} G = \sum_{i=1}^N \lambda_i = 1$)

  (3) calculate the EV entropy $H(G) = -\sum_i \lambda_i \log \lambda_i$ with $0 \cdot \log 0 = 0$

  (4) $VS = \exp(H(G)) \leq N$

VS acts as an effective rank of $\bar{G}$

$\Rightarrow$ properties: – if all points are far from each other: $G_{i,i'} \approx 0$ for $i \neq i'$

$\Rightarrow VS = N$ (maximum diversity)

– if all points are equal: $G_{i,i'} = \frac{1}{N}$ for all i, i'

$\Rightarrow VS = 1$ (minimum diversity)

choosing kernel bandwidth correctly is critical to avoid the extreme cases

[possibly, you can cheat with this]