Gen Hack 2022

Generative Modelling Challenge

2st tutorial session

2022, November 14th

MERCATOR OCEAN INTERNATIONAL

BNP PARIBAS
The bank for a changing world

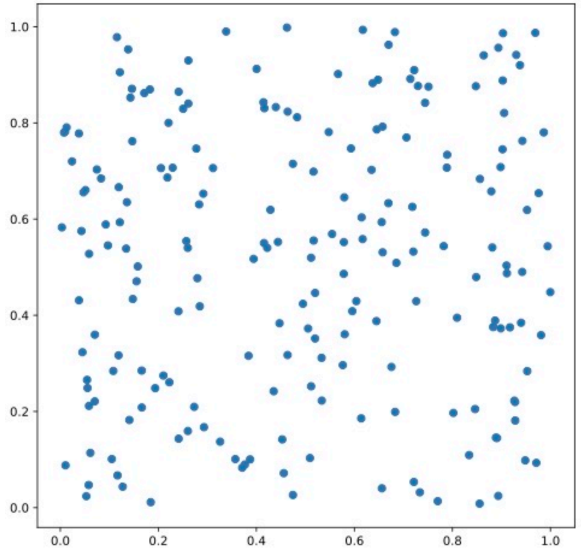
FONDATION ÉCOLE POLYTECHNIQUE

ÉCOLE POLYTECHNIQUE

IP PARIS

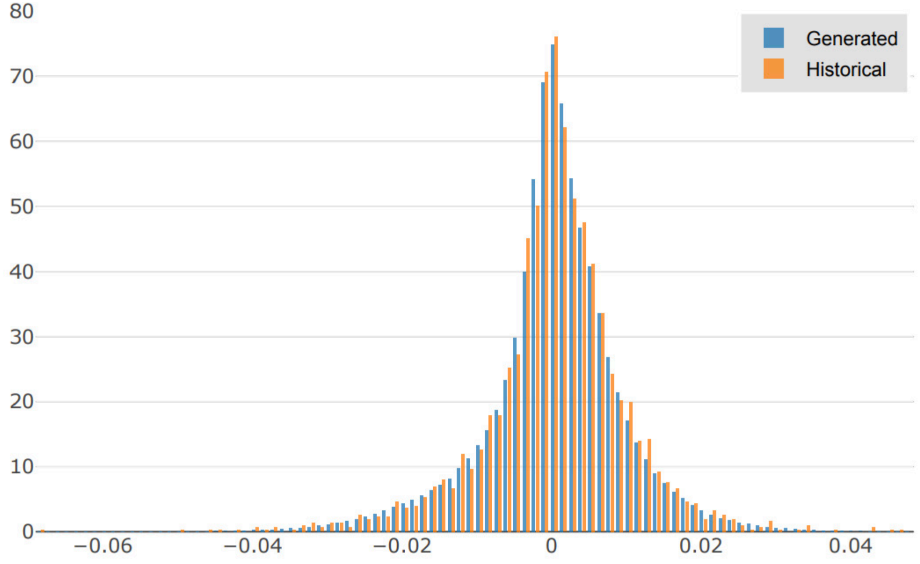
Recap from last tutorial:

Noise



Generator

Data « histograms »



Questions:
- which type of latent noise (and latent distribution)?
- how to build and to learn the Generator?
- which distance, or divergence, or loss function with sampled data and training data?

Today:
- Any random object can be sampled from a single uniform distribution
- But we need a reasonable trade-off between noise and generator complexity

Generator in dimension 1

Gen Hack 2022

Denote by F the c.d.f. of the scalar random variable X.

Theorem (Von Neumann, 1947)

If U has the uniform distribution on $[0,1]$, then

$$F^{-1}(U) \overset{\text{(distr.)}}{=} X.$$

Conversely, if F is continuous, then $F(X) \overset{\text{(distr.)}}{=} \mathcal{U}([0,1])$.

Proof: $P(F^{-1}(U) \leq x)$
$= P(U \leq F(x))$
$= F(x)$. QED.

- ✓ A natural generator of X is its quantile:
  $Q(u) = F^{-1}(u) = \inf\{x: F(x) \geq u\}$
- The Generator may be not a smooth function.
- ✓ One uniform sample ⟷ one X with the same dimension
- ✓ One could take other latent continuous distribution (Gaussian….)
- ✓ How to generalize to larger dimension?

Generator in dimension d

Here the random variable X has d coordinates.

Theorem (Knothe-Rosenblatt rearrangement)

1. Sample $X_1 = Q_{X_1}(U_1)$
2. Sample $X_2 = Q_{X_2|X_1}(U_2)$
3. $\ldots$
4. Sample $X_d = Q_{X_d|X_1,\ldots,X_{d-1}}(U_d)$

✓ The generator looks like $\varphi(u) = \begin{pmatrix} \varphi_1(u_1) \\ \varphi_2(u_1, u_2) \\ \vdots \\ \varphi_d(u_1, \ldots, u_d) \end{pmatrix}$

where the coordinate φ_i is increasing in u_i .

✓ d uniforms $\longleftrightarrow$ d-dimensional X

It is necessary to have the dimension of the latent space equal to the dimension of output variables?

One « uniform » r.v. is enough to generate 2 « uniforms »

Proposition (Dyadic expansion)

For all $t \in [0,1)$, we can write

$$t = \sum_{n \geq 1} \frac{\varepsilon_n(t)}{2^n}, \qquad \text{with } \varepsilon_n(t) \in \{0,1\}$$

and the ε_n are not all equal to 1 after some rank.
The decomposition is unique.

Examples:
0.63804133480990… = 0.1010001101010 …
0.33597927819669… = 0.0101011000000 …

Theorem

For $n \geq 1$ and $U \overset{\text{(distr.)}}{=} \mathscr{U}(0,1)$, $\varepsilon_n(U) \overset{\text{(distr.)}}{=} \mathit{Bernoulli}(\frac{1}{2})$ and the $(\varepsilon_n(U))_n$ are independent.

U_1 and U_2 are two independent and uniform random variables !!

Expand $U = 0.\varepsilon_1\varepsilon_2\varepsilon_3\varepsilon_4\varepsilon_5\varepsilon_6\varepsilon_7\varepsilon_8 \ldots$

$U_1 = 0.\varepsilon_1\varepsilon_3\varepsilon_5\varepsilon_7 \ldots$

$U_2 = 0.\varepsilon_2\varepsilon_4\varepsilon_6\varepsilon_8 \ldots$

Excellent !! But there must be a trap…

Corollary

For any random variable $X \in \mathbb{R}^d$, there is a measurable $\phi : [0,1] \mapsto \mathbb{R}^d$ such that

$$\phi(U) \overset{\text{(distr.)}}{=} X.$$

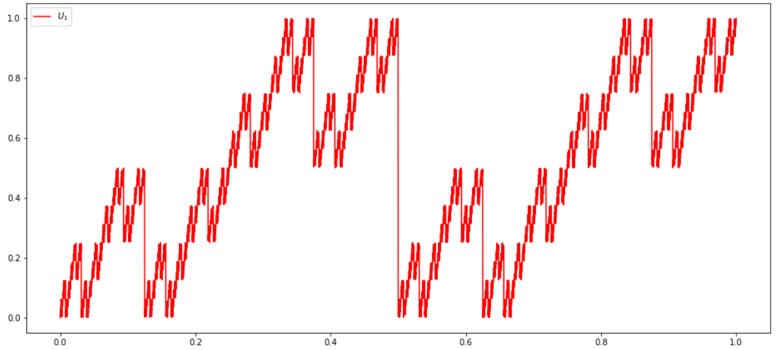
The result holds for general (complete) metric space (and not only R^d) → Kuratowski theorem

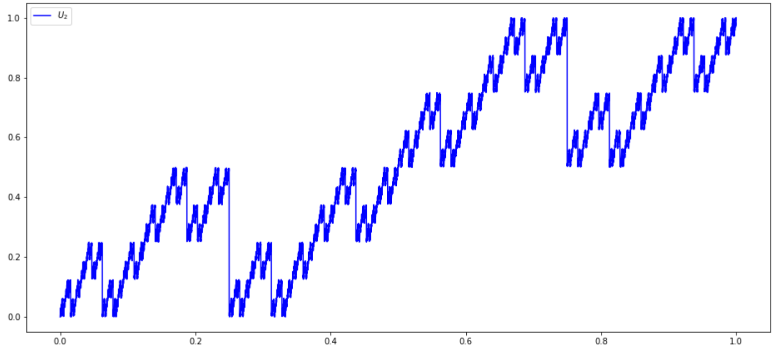
Plots of the functions

$U \to U_1$

and

$U \to U_2$



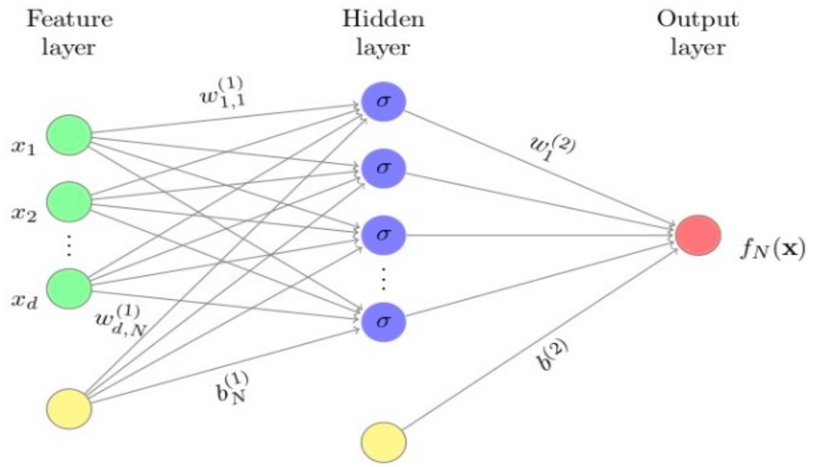


The generator as a function can be ugly ! And thus, hard to approximate and learn.

The one with more inputs (Knothe-Rosenblatt) was likely smoother.

Better to enlarge the latent space.

Using neural networks for approximation



$$f_N(\mathbf{x},\theta) := b^{(2)} + \sum_{k=1}^{N} w_k^{(2)} \sigma\left(\sum_{j=1}^{d} w_{j,k}^{(1)} x_j + b_k^{(1)}\right)$$
$$= b^{(2)} + \sum_{k=1}^{N} w_k^{(2)} \sigma\left(\mathbf{w}_k^{(1)} \cdot \mathbf{x} + b_k^{(1)}\right),$$

$\boldsymbol{\sigma}$ *is the activation function* $(ReLU, \ldots)$

Theorem

Then, there exists a neural network NN (ReLU) with N neurons such that

$$\int_{\mathbb{R}^d} |\varphi(z) - \mathrm{NN}_N(z)|^p \, \mu_Z(dz) \underset{N\to\infty}{\longrightarrow} 0.$$

- ✓ Here, the generator $\boldsymbol{\varphi}$ has no specific regularity (just p-times integrable)
- ✓ But the (theoretical) convergence rate depends on the ratio between its regularity and the dimension (curse of dimensionality) [see Mairov 1998, Pinkus 1999, …]
- ✓ Shallow NN can't beat Deep ReLU NN for smooth functions: see Telgarsky 2015 Yarotsky 2017…

Take-home messages

- ✓ In theory, taking the latent space of dimension 1 (uniform or Gaussian distribution)
- ✓ But the generator may be really complicate
- ✓ Ability of NN to approximate any (smooth or not) functions
- ✓ However, convergence rate depends on the regularity and dimension.
- ✓ Hard to guess what is the right dimension of the latent space
- ✓ Deep NN enjoys better convergence properties (for a fixed size of inputs)

Next tutorials: on effective parametrization of generators

- ✓ Variational Auto Encoder (VAE) and normalizing flows
- ✓ Generative Adversarial Networks (GAN)