GenHack2 - Hackathon for Generative modeling : Simulation of global warming Sea Surface Temperatures

November 17, 2022

1 Introduction

1.1 Context

Dispersed ocean sensors, called stations, help to collect sea temperatures data. However measuring the temperatures on a fine mesh is too expensive for obvious cost reasons. Thus, developing a model able to provide a fine-scale temperature map and able to enrich the possible scenarios is of major importance.

1.2 Objective

Given a vector of input noise vector $Z \in \mathbb{R}^{d_z}$, you have to build a generative model G_θ, parametrized by θ, that can simulate realistic samples

$$\widetilde{X} := G_\theta(Z),$$

similar to a multivariate real climate variable of interest $X \in \mathbb{R}^d$ at $d = 6$ stations.
The objective is to simulate potential future temperature values (**drawn at random during the year**) with spatial dependence between stations.

2 Data

2.1 Dataset description

We consider the daily sea surface temperature (SST) in Kelvin from 1981-09-01 to 2016-12-31 (12,541 days) at several stations.

2.1.1 Train-test split

We consider the training period from 1981-09-01 to 2007-12-31 (n_{train} =9618 days) and the testing period from 2008-01-01 to 2016-12-31 (n_{test} =3288 days).

2.1.2 Data processing

The **seasonality** at each station **in the training dataset** was removed.

2.2 Latent variable

The latent variable $Z \in \mathbb{R}^{d_z}$ will be a **standard normal random vector**, (*i.e.* all the components are independent and zero-mean unit-variance normally distributed random variable).

3 Evaluation

At each evaluation, we will evaluate your model

$$Z \in \mathbb{R}^{d_z} \mapsto G_\theta(Z) \in \mathbb{R}^6$$

which must have the following structure

$$G\left(\begin{bmatrix} Z_1 \\ \vdots \\ Z_{d_z} \end{bmatrix}\right) = \begin{bmatrix} \widetilde{X}_{\text{S1}} \\ \vdots \\ \widetilde{X}_{\text{S6}} \end{bmatrix},$$

with a common and unknown random vector Z with $d_z \leq 50$. Teams will be judged on the test set with $n_{\text{test}} = 3288$ data based on two evaluation score metrics: a marginal and a dependence error.

3.1 Marginals - Anderson-Darling

The Anderson-Darling distance (see [1, 2] for more details) computes a weighted square difference between the hypothetical cumulative distribution function (c.d.f) F from which samples have been drawn, and the empirical c.d.f $\widehat{F}_n$ based on n observations:

$$W_n = n \int_{-\infty}^{\infty} \frac{\left(\widehat{F}_n(x) - F(x)\right)^2}{F(x)(1 - F(x))} dF(x).$$

For this challenge, let $\widehat{F}^s_{n_{\text{test}}}$ be the **testing** empirical distribution function associated to the observations $X^s_1, ..., X^s_{n_{\text{test}}}$ and let $X^s_{1,n_{\text{test}}} \leq \cdots \leq X^s_{n_{\text{test}},n_{\text{test}}}$ be the order statistics for each station $s = 1, \ldots, 6$.

We denote $\widetilde{u}^s_{i,n_{\text{test}}}$ the model probability of a generated variable $G_\theta(Z) = \widetilde{X}$ for a specific station s such that

$$\widetilde{u}^s_{i,n_{\text{test}}} = \frac{1}{n_{\text{test}}+2}\left(\sum_{j=1}^{n_{\text{test}}} \mathbb{1}\left\{X^s_j \leq \tilde{X}^s_{i,n_{\text{test}}}\right\} + 1\right).$$

The Anderson-Darling distance for each station s can be computed as

$$W^s_{n_{\text{test}}}(Z) = -n_{\text{test}} - \frac{1}{n_{\text{test}}}\sum_{i=1}^{n_{\text{test}}}(2i-1)\big(\log(\widetilde{u}^s_{i,n_{\text{test}}}) + \log(1-\widetilde{u}^s_{n_{\text{test}}-i+1,n_{\text{test}}})\big). \tag{1}$$

Then, the global metric on the marginals is just the average distance for all stations and is computed as

$$\mathcal{L}_M(Z) = \frac{1}{d}\sum_{s=1}^{d} W^s_{n_{\text{test}}}.$$

3.2 Dependence - Absolute Kendall error

Kendall's dependence function (see [3] for more details) characterizes the dependence structure associated with a copula C and is the univariate cumulative distribution function defined by $K_C(t) = \mathbb{P}\left(C\left(U^{(1)},\dots,U^{(d)}\right) \leq t\right)$ for all $t \in [0,1]$ and $\left(U^{(1)},\dots,U^{(d)}\right)$ a random vector with uniform margins on $[0,1]$. The estimation of the Kendall's dependence function is based on the **testing** pseudo-observations

$$R_i = \frac{1}{n_{\text{test}}-1}\sum_{j\neq i}^{n_{\text{test}}} \mathbb{1}\left\{X^1_j < X^1_i, \dots, X^d_j < X^d_i\right\},$$

and we consider equivalently the ones from the model

$$\widetilde{R}_i = \frac{1}{n_{\text{test}}-1}\sum_{j\neq i}^{n_{\text{test}}} \mathbb{1}\left\{\widetilde{X}^1_j < \widetilde{X}^1_i, \dots, \widetilde{X}^d_j < \widetilde{X}^d_i\right\}.$$

Then, the dependence metric that we define can be computed as a L^1 norm

$$\mathcal{L}_D(Z) = \frac{1}{n_{\text{test}}}\sum_{i=1}^{n_{\text{test}}}\left|R_{i,n_{\text{test}}} - \widetilde{R}_{i,n_{\text{test}}}\right|,$$

where $R_{1,n_{\text{test}}} \leq \dots \leq R_{n_{\text{test}},n_{\text{test}}}$ (resp. $\widetilde{R}_{1,n_{\text{test}}} \leq \dots \leq R_{n_{\text{test}},n_{\text{test}}}$) are the order statistics associated with $\{R_1,\dots,R_{n_{\text{test}}}\}$ (resp. $\{\widetilde{R}_1,\dots,\widetilde{R}_{n_{\text{test}}}\}$).

Ranking. The total evaluation score for each metric will be computed on several noises samples $Z^1,\dots,Z^M$ in order to reduce the statistical error such that

$$\mathcal{L}_M = \frac{1}{M}\sum_{m=1}^{M}\mathcal{L}_M(Z^m), \quad \mathcal{L}_D = \frac{1}{M}\sum_{m=1}^{M}\mathcal{L}_D(Z^m).$$

4 General rules

- All your code must be in Python 3.8.
- Git will be used to push your model and Docker to build an environment to evaluate your code. You don't have to master git and Docker, just have them installed in your local machine.
- No restriction on how you train your model, but before each submission date you must ensure that your model runs in our predefined framework (see the Paragraph Submission below).

Requirements.

- Python 3.8
- Git
- Docker

Create your private repository.

1. Visit the GenHack2 git repo where you will find:
   - `data/`: folder containing the training data (`data_train.csv`) and an example of a noise file (`noise.npy`). Feel free to use another noise for training your model but keep in mind that
     - d_z must be less or equal to 50,
     - you will be evaluated on a common and unknown standard normal random vector Z.
   - `requirements.txt`: text file containing the libraries with their associated versions you used in the `model.py` file. **Do modify ✓**
   - `Dockerfile`: docker image in order to create a container. **Do not modify ✗**
   - `main.py`: main python file containing the simulation function. **Do not modify ✗**
   - `model.py`: python file containing your generative model and for loading the parameters. **Do modify ✓**
   - `parameters/`: folder where you must put the parameters of your model. **Do modify ✓**
   - `run.sh`: bash script to run your simulation. **Do not modify ✗**
2. Duplicate the repository (just one of the team member)
   (a) create an empty **private** repository with as repo name the **NAME_OF_YOUR_TEAM**. Be sure that your 'base' branch is called `master`.
   (b) invite your team members and **generative-hackathon**
   (c) open a terminal where you want to put your private repo and run the following commands.
   Note: you can either clone on HTTPS (default) or on SSH depending on your setting.

```
$ git clone --bare
    https://github.com/generative-hackathon/GenHack2.git (HTTPS)
OR
git clone --bare git@github.com:generative-hackathon/GenHack2.git (SSH)
$ cd GenHack2.git
$ git push --mirror
    https://github.com/YOUR_GIT_USERNAME/NAME_OF_YOUR_TEAM.git (HTTPS)
OR
git push --mirror
    git@github.com:YOUR_GIT_USERNAME/NAME_OF_YOUR_TEAM.git
$ cd ..
$ rm -rf GenHack2.git
$ git clone https://github.com/YOUR_GIT_USERNAME/NAME_OF_YOUR_TEAM.git
    (HTTPS)
OR
git clone git@github.com:YOUR_GIT_USERNAME/NAME_OF_YOUR_TEAM.git
```

Submission.

1. open Docker
2. put in your local repo your latest version of `models.py` and `parameters/` and push them to your git repo
3. open a terminal where your local repo is, and run the command

   ```
   $ sh run.sh
   ```

   This command will

   (a) pull your git repo - so again, be sure that your git repo is up to date
   (b) run your code in a docker container
   (c) push to your git repo a `check.log` file containing a debug message

4. open the file `check.log` to see if you had a ”successful simulation”. If not, fix the error and run again

   ```
   $ sh run.sh
   ```

References

[1] T.W Anderson and D.A. Darling. Asymptotic theory of certain” goodness of fit” criteria based on stochastic processes. *The Annals of Mathematical Statistics*, 193–769212, 1952.

[2] T.W Anderson and D.A. Darling. A test of goodness of fit. *Journal of the American Statistical Association*, 765–769, 1954.

[3] C. Genest and L.-P. Rivest. Statistical inference procedures for bivariate Archimedean copulas. *Journal of the American Statistical Association*, 88(423):1034–1043, 1993.