Projet LINMA1731

Philémon Beghin, Jehum Cho

25/02/2022

The Lorenz system is a system of ordinary differential equations notable for having chaotic solutions for certain parameter values and initial conditions. In particular, the Lorenz attractor is a set of chaotic solutions of the Lorenz system. In popular media the "butterfly effect" stems from the real-world implications of the Lorenz attractor, i.e. that in some physical system, in the absence of perfect knowledge of the initial conditions (even the minuscule disturbance of the air due to a butterfly flapping its wings), our ability to predict its future course will always fail.[1] Figure 1 shows a representation of the Lorenz system [1].

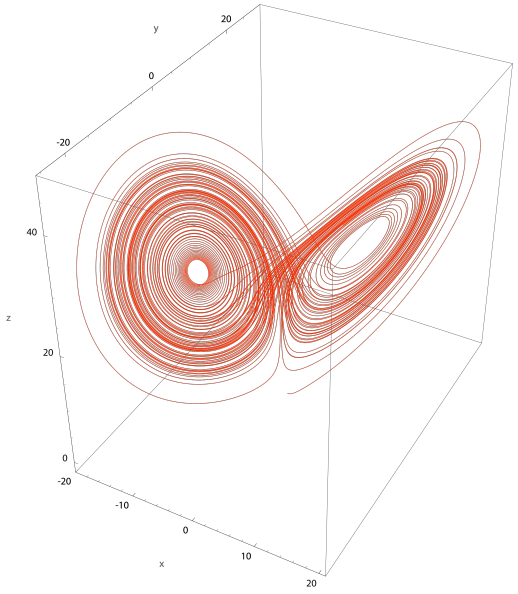

Figure 1 – Lorenz attractor

In this project, you will be asked to study the influence of some parameters on the Lorenz system and implement a particle filter that could estimate the true position of a point in the system based on noisy observations.

1. `https://fr.wikipedia.org/wiki/Attracteur_de_Lorenz`

1 Theoretical Questions

The Lorenz system is ruled by the following state-space model:

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases} \tag{1}$$

where x, y and z stand for the position coordinates of the point in the system and σ, ρ and β are positive real parameters. Lorenz used the values $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$ for which the system shows a chaotic behaviour.

1.1 Empirical probability density function (PDF)

We consider the trajectory of a point starting at $(x, y, z) = (1, 1, 1)$ at time $t = 0$. We will consider the evolution of its dynamic every 0.02 s for a total duration of 100 s. We work in the spatial domain $[-20, 20] \times [-30, 30] \times [0, 50]$.

a) First, you are asked to study the probability density function of a point inserted in the system at $(x, y, z) = (1, 1, 1)$. In order to compute the empirical PDF, you will have to divide the spatial domain into 3D cubic boxes of length $l = 5$ and determine how many times the point go into each box during the considered time interval (e.g. How many times did the point go into the box $[0, 5] \times [0, 5] \times [0, 5]$ during 100 s?). The graph of the PDF you will derive is in 4 dimensions (x, y, z positions of the boxes and the value of the PDF inside these boxes). Represent it using projections on xy, yz and xz spaces (under the form of "colored" matrices or 3D histograms). Comment your figures.

b) Propose at least two ways to measure the distance between two statistical distributions P and Q (i.e. measuring the distance between their histograms). Comment their relevance. This question is deliberately open. The keyword *statistical distance* might help you in your research.

c) Study the impact of the parameters σ, ρ and β on the distribution. Then, using your two proposed methods, compute the distance between the first PDF and a new one obtained with parameters $\sigma = 5$, $\rho = 18$ and $\beta = 8$. Comment your results.

d) Study the impact of the initial conditions on the distribution. Then, using your two proposed methods, compute the distance between the first PDF and a new one obtained with initial conditions $(x, y, z) = (10, 10, 10)$. Comment your results.

Practical details related to the implementation:

1. You are provided with the *Project_student.py* file that simulates the Lorenz system. If you run this file, one figure will appear. It shows the discretized version of the Lorenz system. The vector $states \in \mathbb{R}^{5000 \times 3}$ stands for the (x, y, z) position of the point every 0.02 s for a total duration of 100 s.
2. We strongly recommend you to install the *Plotly* package to visualize your 3D graphs. The code plots the graph using *matplotlib* to avoid packages conflict, but the *Plotly* version is available in the code. You just have to uncomment the dedicated section once you have installed the package. Anaconda users can type in their terminal one of the lines given at `https://anaconda.org/conda-forge/plotly` to install the package.

1.2 Particle Filter

Consider a dynamic state-space model:

$$\begin{aligned} \mathbf{x}_t &= f(\mathbf{x}_{t-1}, v_t) \qquad \text{(state equation)} \\ \mathbf{y}_t &= g(\mathbf{x}_t, w_t) \qquad \text{(measurement equation)} \end{aligned} \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ denotes the states and $\mathbf{y} \in \mathbb{R}^{n_y}$ the observations. v_t and w_t are the process noise and the measurement noise respectively. Let us assume that $p(\mathbf{x}_0)$ is the prior distribution at $t = 0$.

In applications, it is of interest to estimate $\varphi(\mathbf{x}_{0:t})$ with the observations available until that point $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t\}$. This can be done by using the posterior density $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. The issue is that often the case this posterior density is not readily available, so we need to approximate it. Particle Filter approximates the posterior distribution by the empirical estimate of the samples of this distribution so-called particles. Over time particle filters might face the weight degeneracy issue. Practically, we observe low levels of importance weights from a large number of samples and a small number of samples dominate the others.

Resampling

In order to circumvent the degeneracy issue, we can use resampling for eliminating samples with low importance weights and multiplying samples with high importance weights. Resampling corresponds to the mapping from $\{\tilde{\mathbf{x}}_t^{(i)}, \tilde{\omega}_t^{(i)}\}$ into $\{\mathbf{x}_t^{(i)}, 1/N\}$. Here $\mathbf{x}_t^{(i)} = \tilde{\mathbf{x}}_t^{(j)}$ with some index j. Equivalently, this process can be seen as determining the number $N_t^{(i)}$ of sample $\tilde{\mathbf{x}}_t^{(i)}$ for each i. There are several methods to achieve this goal and the performance varies. Compare and analyze the following resampling methods by obtaining a performance measure $var(N_t^{(i)})$ in a closed form and the complexity of the algorithm.

1. (Multinomial Resampling) Determining $N_t^{(i)}$ from a multinomial distribution with parameters N and $\tilde{\omega}_t^{(i)}$.
2. (Residual Resampling) First set $\tilde{N}_t^{(i)} = \lfloor N\tilde{\omega}_t^{(i)} \rfloor$, and draw a sample from a multinomial distribution with parameters $N - \Sigma_{i=1}^{N} \tilde{N}_t^{(i)}$ and $\tilde{\omega}_t^{(i)}$. $N_t^{(i)}$ is the addition of $\tilde{N}_t^{(i)}$ and the result of the sample from the multinomial distribution.
3. (Systematic Resampling) Sample a set of N points in the interval [0,1], whose distances are exactly 1/N apart. Determine $N_t^{(i)}$ by counting the number of points lying between $\Sigma_{j=1}^{i-1} \tilde{\omega}_t^{(j)}$ and $\Sigma_{j=1}^{i} \tilde{\omega}_t^{(j)}$.

2 Particle Filter Implementation for the Lorenz System

2.1 Data Generation

Consider the Lorenz system again. A nonlinear continuous-time state-space description is:

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases} \tag{3}$$

where the values of the parameters are $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$ for which the system shows a chaotic behaviour.

Our goal is to estimate the location of a point that is following the underlying Lorenz system. We observe the trajectory of the point every 0.02 seconds, and the overall measuring time is 100 seconds. We can observe all of the three coordinates, but with some measurement noise $w \sim \mathcal{N}(0, \sigma^2_{mes})$. Let us assume that the measurement noises for each coordinate are independent from each other. In order to generate a "true" trajectory, use the function *odeint* in the python library *scipy*, as given in the example code (*Project_student.py*). The initial point should be $(x(0), y(0), z(0)) = (1, 1, 1)$. With the obtained trajectory, generate the observation vector $\mathbf{y}_{1:t}$ with $\sigma^2_{mes} = 1$.

2.2 Discretization of Continuous-Time State-Space Equation

Let $\mathbf{x}_t = [x(t), y(t), z(t)]^T$. The continuous-time state-space equations (3) can be discretized using the Runge-Kutta method to obtain the following discrete-time nonlinear state-space system of equations:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{x}_{t-1} + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4) + v_t \\ \mathbf{y}_t &= \mathbf{x}_t + w_t \end{aligned} \tag{4}$$

where

$$f(\mathbf{x}_t) = \begin{bmatrix} \sigma(y(t) - x(t)) \\ x(t)(\rho - z(t)) - y(t) \\ x(t)y(t) - \beta z(t) \end{bmatrix}, \tag{5}$$

$f_1 = f(\mathbf{x}_t)$, $f_2 = f(\mathbf{x}_t + h\frac{f_1}{2})$, $f_3 = f(\mathbf{x}_t + h\frac{f_2}{2})$, $f_4 = f(\mathbf{x}_t + hf_3)$ and $h = 0.02s$ is the sampling period. The terms v_t and w_t denote the disturbance/noise added to the states and output, respectively. While w_t is multivariate version of the measurement noise w from above, v_t is added because of the inaccuracy of the discretization approximation. There exists a more systematic approach to estimate v_t; however, this is out of scope for our project. Here, we assume that v_t follows Gaussian distribution, namely $\mathcal{N}(0, \Sigma)$.

2.3 Particle Filter Implementation

Implement the Generic Particle Filter of the slides using the SIR proposal density. Use the discretized state-space equation (4) and the data generated in chapter 2.1. Test it with different options including as follows.

1. Σ for the approximation error. (e.g. what happens if we assume that there is no error? or if Σ is too big?)

2. The frequency of the observation h. (What happens if h is smaller or bigger than 0.02s)
3. Resampling methods. (Test all of the three resampling methods without using a certain python library. Is any method better than the other? If so, why?)

Report graphs to illustrate:

1. The real state trajectory (the data you generate)
2. The estimated filtered states
3. Distance between the real trajectory and the estimated one as a function of t.

All results must be properly analyzed and commented. Notice that the report without proper analysis of the results will not be able to get a high grade.

Practical details

Modality	The project is carried out by groups of two students. If you need to work alone or do not know anybody to work with, please contact us to find an arrangement. Each group must register on Moodle by Friday 11 March 2022, 23.59 pm.
Supervision	Office hours from week 7 to week 13: one hour of permanence on Tuesday 3 pm at the Euler building (room A.007) by Philémon Beghin and Friday 2 pm on Teams by Jehum Cho. For the office hours on Teams, we recommend you to send a clear description of your questions during the office hour (not earlier or later), and wait for the teaching assistant to respond you back. Please note that there can be several groups waiting for one teaching assistant, so that you might experience some delay.
Report	English is strongly recommended. But the course is French friendly, hence French is allowed without penalty. However, reports in French, if any, will have to go through a different grading procedure. The goal is not to evaluate your English skills and we will therefore not pay attention to the quality of the language, but to the scientific quality of your report instead. Maximum 10 pages for the total of both parts (e.g. 5 pages each).
Deadlines	Part I : Friday 22 April 2022 at 18.15 pm on Moodle. The report (in pdf format) and the code (in Python .py or .ipynb) will be submitted together in a zip file named `LINMA1731_2022_Project_Part1_NAME1_NAME2.zip`. Part II : Friday 13 May 2022 at 18.15 pm on Moodle. The report (in pdf format) and the code (in Python .py or .ipynb) will be submitted together in a zip file named `LINMA1731_2022_Project_Part2_NAME1_NAME2.zip`.

References

[1] `https://en.wikipedia.org/wiki/Lorenz_system`