LINMA1731 - Project 2020

Aircraft localization

In this project you are going to analyse flying objects. In a first part, you will determine the nesting preferences of birds, and do some theoretical analysis of multivariate normal distribution. Then, in a second part you will localize a plane flying above the belgian Ardennes, using a simple state model and an elevation map.

Part I - Estimators and multivariate Gaussians

1. MAP estimator

In this section, you are going to analyse birds in your garden via a MAP estimator.

Question 1.1 *Bird experts have determined that the probability θ that a bird goes to nesting box of type A instead of nesting box of type B follows a beta distribution: $\theta \sim \mathcal{B}(2,4)$ (a priori knowledge). You observed N birds in your garden and noted to which nesting box they were going (you noted "True" if the bird was going to nesting box of type A and "False" otherwise). Each bird is going to only one nesting box. Determine the Maximum A Posteriori (MAP) estimator of θ, the probability to go to nesting box A for birds in your garden.*

Question 1.2 *What can you say about this estimator? What is the bias, the mean square error? Is the estimator consistent and asymptotically normal (these two characteristics must be considered in the Fisher sense, as in your slides)?*

For this question you may need the following properties:

1. $\mathbb{E}\{g(Z,\Theta)\} = \int_\theta \mathbb{E}[g(Z,\Theta)|\theta] f_\Theta(\theta)\, d\theta$
2. $Pr[g(Y-X) \geq \epsilon] \leq \frac{\mathbb{E}\{g(Y-X)\}}{g(\epsilon)}$
3. Central limit theorem: $\frac{\sqrt{N}(\bar{X}_n - \mu)}{\sigma} \xrightarrow[N\to\infty]{\mathcal{D}} \mathcal{N}(0,1)$,
   where $\bar{X}_n = \frac{\sum_{i=1}^N X_i}{N}$, μ is the mean of X and σ its standard deviation.

Question 1.3. *Compute now the Maximum Likelihood (ML) estimator of this problem. Compare the MAP with the ML estimators in terms of (Bayesian) bias and mean square error.*

Question 1.4. *Implement your MAP and ML estimators and apply them on the data on moodle named* `Estimators1`*. What are the estimated values of θ? You also have data about birds in the garden of your friend (*`Estimators2`*). Apply your estimator on these data as well. Comment the results you observed.*

2. Sampling from Multivariate Normal Distributions

In this section, we focus on a method to sample a random variable from a multivariate normal distribution using the one-dimensional standard normal distribution.

Question 2.1. *Show that affine transformations of the multivariate normal distribution follow the multivariate normal distribution. Specifically, let* $X \sim \mathcal{N}(\mu_X, \Sigma_X), Y = LX + u$*, where* L *is a linear map and* u *is a vector in the same space with* Y*. Show that* Y *also follows a multivariate normal distribution and identify its mean and variance. Consider only when dimensions of* X *and* Y *are the same.*

Now, let's say the distribution from which you want to sample is $\mathcal{N}(\mu_Y, \Sigma_Y)$.

Question 2.2. *Let* $X \sim \mathcal{N}(0, I_d)$*. Show that* $Y = LX + \mu_Y$ *follows* $\mathcal{N}(\mu_Y, \Sigma_Y)$*, where* $\Sigma_Y = LL^T$ *with a lower triangular matrix* L*.*

Note that we can sample X in the Question 2.2 using the univariate standard normal distribution since each variable in X is independent from the other variables. The decomposition $A = LL^*$ is called the Cholesky Decomposition. When A is Hermitian and positive semi-definite then the decomposition exists and if A is symmetric then L is real, which implies $L^* = L^T$. Notice that every covariance matrix of a real random vector is indeed positivie semi-definite and symmetric, so in our case L always exists and its components are real. [1]

Part II - Particle filtering

Consider you are in an airplane flying above the belgian Ardennes, that has no information about its precise position. However, it knows its speed and the onboard sensors can provide at any time noisy measurements of the ground elevation below the plane y_t (via the altitude of the plane, and its distance to the ground). Given a map of the ground elevation $h(\cdot)$, you should be able to localize the airplane using particle filtering.

State model

Suppose that the motion of the plane can be described using this simple state model:

$$x_{t+1} = x_t + v_t\delta_t + w_t$$
$$y_t = h(x_t) + e_t$$

where x_t is the position of the plane, δ_t is the time interval between 2 measurements, y_t is the ground elevation computed by the plane and w_t, e_t are Gaussian noises.

On moodle you will find an elevation map of the Ardennes called `Ardennes.txt` that contains at each line the longitude, latitude and ground elevation of a point. You are also given in file `elevationMap.py` an implementation of the function $h(\cdot)$ that you have to use in this project. Note that this function takes in argument a number in $[0, 1]$ or a vector in $[0, 1] \times [0, 1]$.

Particle filtering

- Suppose, as a first step, that the plane moves along a straight line at fixed altitude (see figure 1). $x_t, v_t = 1.6$ and $w_t \sim \mathcal{N}(0, 0.004)$ are then scalars, and $e_t \sim \mathcal{N}(0, 16)$. The initial position of the aircraft is unknown, but you can use as initial guess a uniform distribution on the domain $x_1 \sim \mathcal{U}(0, 1)$. Implement a bootstrap particle filter to localize the plane from measurements of its onboard sensors.

  On moodle, you will find a dataset called `measures1D`. There are two components in the dataset. First, Y_t is the estimation of the ground elevation, measured by the airplane at a frequency of 100 measurements/seconds. You will utilize this data for the implementation and get the estimate of your position. Second, $POSITION_t$ is the actual position which is

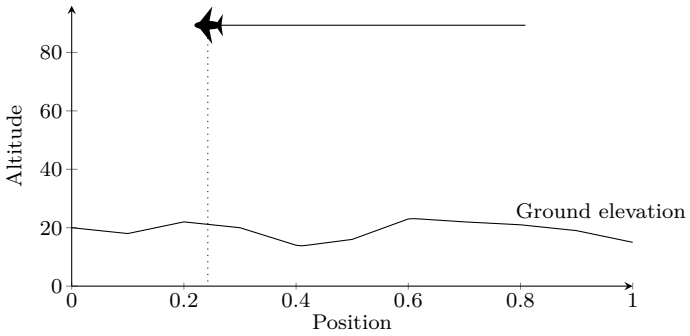

FIGURE 1 – Plane moving along a straight line.

supposed to be unknown, this is only for tracking the error between the actual location and the estimate of the location from your particle filter. Test your particle filter on this dataset with various numbers of particles (iterate resampling 50 times: $T = 50$), and observe the evolution of the PDF estimated by the particles. Report a graph of the error vs time. The error is the difference between the estimated position and the true position. Comment your results.

- Suppose now that the plane can move freely in a 2-dimensional space, but still at fixed altitude. $x_t, v_t = [-0.4, -0.3]$ and $w_t \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, 10^{-5} \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix} \right)$ are then vectors with 2 elements, and $e_t \sim \mathcal{N}(0, 16)$. This time let's say we have a bit more information of our initial location. You can again guess the initial position of the aircraft using a uniform distribution on the domain $x_1 \sim \mathcal{U}([0, 1] \times [0, 1])$. Implement a bootstrap particle filter to localize the plane from measurements of its onboard sensors.

  Note that the noise w_t follow in this case a multivariate normal distribution. Here, you need to use the method described in the subsection 2 of part 1 to generate a multivariate normal distribution in your implementation. Design a function whose arguments are a mean vector and a covariance matrix, that returns a sample of a multivariate normal distribution using only the one dimensional standard normal distribution. Observe that the Cholesky decomposition can be done very easily for two-dimensional cases. You will not need any packages for the decomposition.

  On moodle, you will find a dataset called `measures2D`. There are three components in the dataset. First, Y_t is the estimation of the ground elevation, measured by the airplane at a frequency of 100 measurements/seconds. $POSITION\_X1_t$ and $POSITION\_X2_t$ are coordinates indicating the actual position of the plane through time. You should not include this data into your implementation, but you will need them to calculate the difference between the actual location and the estimate of the location computed by your particle filter. When you calculate the distance, use the Euclidean distance. Test your particle filter on this dataset with various numbers of particles (iterate resampling 100 times: $T = 100$), and observe the evolution of the PDF estimated by the particles. Report a graph of the error with time steps. Comment your results.

Practical details

Teaching assistants	Jehum Cho and Cécile Hautecœur
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 13 March 2020, 6.15 pm.
Supervision	Supervised session on week 8 (usual schedule for exercises sessions, Euler building a.002) and office hours from week 10 to week 13: on Monday 10 am (Euler building, office a.011) and on Thursday 2 pm (Core building office B.127), one hour of permanence.
Report	The course is French friendly, hence French is allowed without penalty. However, if you write your report in French, then it will have to go through a specialized grading process (without penalty). English is strongly recommended. 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. Please write 10 pages maximum (in total, for both reports).
Deadline	Part I : Friday 24 April 2020 at 6.15 pm on Moodle platform. For the mid-term deadline, you just need to submit the report of the first part and not the code (in pdf, with the filename `LINMA1731_2020_Project_Part1_NAMEstudent1_NAMEstudent2.pdf`). Part II : Friday 15 May 2019 at 6.15 pm on Moodle platform. The report (in pdf format) and the code (in Python) will be submitted together in a zip file named `LINMA1731_2020_Project_Part2_NAMEstudent1_NAMEstudent2.zip`. The code consists of your two particle filters and your function that sample from multivariate normal distribution. Please comment your code and indicate clearly what you did.
Evaluation	Evaluation criteria are available on Moodle website in the submission modules.

References

[1] Roger A. Horn, C. R. J. Matrix analysis. *Cambridge University Press* (1985).