Herramientas Computacionales para Ciencias
Homework 12

Mauricio Sevilla*

06/05/2019

Rules

Note Read carefully the complete homework before starting, so you will know what is the results you have to get!.

This week we are going to concentrate on Monte Carlo methods and applications.

On this assignment you will find three different levels (Beginner, Basic and Intermediate), you have to do only one of each one, you can choose which.

There is an additional point that must be developed on class before sending the assignment.

This part must be saved on a jupyter Notebook named as your UniAndes username.

Introduction

There are a lot of different situations where random numbers are relevant, we are going to see some of them.

Beginner

Game Theory: Parrondo's Paradox

The Parrondo's paradox stands that *The combination of loosing strategies, can lead to a winning strategy.*. On this problem we will test that sentence on a particular case which was named after J.M. Parrondo when studying molecular engines.

Suppose that we are going to play two games, and you are going to have bigger probability to win than I do.

Game A

Suppose that we trow a coin, but it is not fair!, you have a probability

$$P(\text{You Win}) = 0.5 + \epsilon \tag{1a}$$

$$P(\text{I Win}) = 0.5 - \epsilon \tag{1b}$$

Game B

The total money I have is multiple of a certain number M.

- If my Capital when we play $C(t)\%M = 0$

$$P(\text{You Win}) = 0.9 + \epsilon \tag{2a}$$

$$P(\text{I Win}) = 0.1 - \epsilon \tag{2b}$$

- If my Capital when we play $C(t)\%M \neq 0$

$$P(\text{You Win}) = 0.25 + \epsilon \tag{3a}$$

$$P(\text{I Win}) = 0.75 - \epsilon \tag{3b}$$

- Implement two functions, one for each game.
  - Use $\epsilon = 0.005$.
  - Use $M = 5$.
- Test that if one plays one or the other game, the probability that I lose is bigger, to do so,

*email=j.sevillam@uniandes.edu.co

  - Do a loop where inside you use your function, and save the results on a list.
  - Plot the lists.
- You will have to make averages so that you can see general behavior, so repeat 5000 times the following.
  - You will play the games randomly mixed, so that select randomly (50% of probability for each game) which game you are going to play at each step.
  - Repeat 200 times.
- Plot the averages of the 5000 repetitions.
- Comment your results.

Random Numbers: Accepting or Rejecting

There are many ways to generate random numbers, one of the most simplest is following this procedure,

- Consider the case that, you want to generate Gaussian random numbers with a given standard deviation $\sigma = 1$ and a mean $\mu = 0$, this Gaussian (Not Normalized so it can not be called probability density) is described by,

$$G(x) = \exp\left(-\frac{1}{2}x^2\right) \tag{4}$$

- In addition, you have to restrict your domain to $x \in (a, b)$, in our case we will choose $a = -b = 3$
- To generate a single point you have to generate a pair of uniform random variables, $x \in (a, b)$ and $y \in (0, 1)$
- if $y < G(x)$ then accept x as your Gaussian distributed point. reject x otherwise.
- plot a histogram of the accepted numbers normalized (Use the `density` option of `plt.hist()`).
- Use the histogram to make a fit to the following function (Use `curve_fit`)

$$G(x) = A\exp\left(-\frac{1}{2}x^2\right) \tag{5}$$

- Comment your results.

Basic

Random Numbers on Sphere: Gaussian

There are several methods to calculate random numbers uniform distributed over the surface of a sphere, they are not as natural as one would expect.

On spherical coordinates, one has as variables (r, θ, φ) where we will use as a convention θ the polar angle and φ the azimuth, such that the surface of a sphere is defined by $r = cte$. One would expect to generate the points uniformly distributed by taking θ and φ as uniform distributed points. Unfortunately, this procedure leads to have concentration on the poles ($\theta \approx 0$ and $\theta \approx \pi$). On this part, we will correct that problem

- Watching the concentration.
  - Generate a set or uniform numbers (~ 1000) for each variable θ and φ.
  - Plot θ vs φ, you may use points or the function `plt.scatter`
  - Convert these coordinates to Cartesian coordinates by using the relations

$$x = r\cos(\varphi)\sin(\theta) \tag{6a}$$

$$y = r\sin(\varphi)\sin(\theta) \tag{6b}$$

$$z = r\cos(\theta) \tag{6c}$$

    Where you can use $r = 1$.
  - Use a scatter function to plot (x vs y), (x vs z) and (y vs z).
  - Comment the differences.
  - You can plot the 3D plot by using:

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_aspect('equal')
ax.scatter(x,y,z)
```

- Solving the concentration.
  - Generate three set of normal distributed points (Gaussian).
  - Take each value of x, y, z and divide by the norm, so that

$$x = \frac{x}{\sqrt{x^2+y^2+z^2}} \tag{7a}$$

$$y = \frac{y}{\sqrt{x^2+y^2+z^2}} \tag{7b}$$

$$z = \frac{z}{\sqrt{x^2+y^2+z^2}} \tag{7c}$$

  - Use the relations (8) to find the corrected (x,y,z).
  - Plot again the pairs (x vs y), (x vs z) and (y vs z).
  - Comment your results
  - Repeat the 3dPlot.

Random Numbers on Sphere: Jacobian

- Watching the concentration.
  - Generate a set or uniform numbers (~ 100) for each variable θ and φ.
  - Plot θ vs φ, you may use points or the function `plt.scatter`
  - Convert these coordinates to Cartesian coordinates by using the relations

$$x = r\cos(\varphi)\sin(\theta) \tag{8a}$$

$$y = r\sin(\varphi)\sin(\theta) \tag{8b}$$

$$z = r\cos(\theta) \tag{8c}$$

    Where you can use $r = 1$.
  - Use a scatter function to plot (x vs y), (x vs z) and (y vs z).
  - Comment the differences.
  - You can plot the 3D plot by using:

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_aspect('equal')
ax.scatter(x,y,z)
```

- Solving the concentration.
  - The main reason for this concentration to happen is that the way distances are measured on Cartesian coordinates, is different than the way they are measured in Spherical coordinates, this correction can be made by using what is called the *Jacobian*.

$$\phi = 2\pi\nu \tag{9a}$$

    Where ν is a uniform random number from 0 to 1. (The same as the previous case)

$$\theta = \arccos(2\mu - 1) \tag{9b}$$

    Where μ is a uniform random number from 0 to 1. (The correction)
  - Use the relations (8) to find the corrected (x,y,z).
  - Plot again the pairs (x vs y), (x vs z) and (y vs z).
  - Comment your results
  - Repeat the 3dPlot.

Random Numbers: Metropolis-Hastings

The Metropolis-Hastings method allows us to find numbers that follows a given distribution, we are going to use it to find Gaussian distributed points.

- Define a function,
$$q(x) = \frac{1}{p(x)} \tag{10}$$
and like we are going to use a Gaussian,
$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{11}$$
- Create a function that receives an integer N and generates the numbers as follows,
  - Initialize the random number on $r = 0$
  - Calculate the function $p = q(0)$ and save it on a variable.
  - Create an empty array-like to save your numbers.
  - Do a loop until N, and inside do
    * Select a random number (Uniform) on an interval (You can take -10 to 10) and save it on r_n
    * calculate the function $p_n = q(r_n)$
    * Evaluate if $p_n > p$ then
      · $p = p_n$
      · $r = r_n$
    * If not, then
      · Generate a random number u from 0 to 1.
      · if $u < (p_n/p)$, then $r = r_n$ and $p = p_n$

    Save r as your Gaussian distributed point.
- plot a histogram of what the function returns.
- Comment your results.

Single Random Walk and Rolling Average

On this part of the Homework you will see how to smooth some signal as one of a random walker to see general behavior.

- Generate a set of $N = 100$ random normal distributed numbers ($\mu = 0$, $\sigma = 1$).
- Do the cumulative sum on a function such that
$$x_i = \sum_{j=0}^{i-1} x_j \tag{12}$$
- compare with the function `np.cumsum`
- create an array-like with the cumulative sum
- Plot it, so that one can say this is the trajectory on time, as you can see it looks noisy.
- Create a linspace x from 0 to 100 (100 numbers)
- Create another array-like with the function $2\sin(x/5)$.
- Add the noisy signal to this sine function.
- Plot the result.
- Implement a rolling average, a moving rolling, given a window w is calculated as
$$x_i = \frac{1}{w} \sum_{j=i-w}^{i} x_j \tag{13}$$
In other words, is the average is done with the previous w numbers.
- Plot the signal and its rolling average.
Hint: Remember that the rolling average has w less numbers.

Intermediate

Random Walks: Diffusion

Random walks can reproduce a diffusion process, to see that one has to take into account more than just one walker,We are going to work on two dimensions.

- As we are going to have more than one random walker, generate two arrays filled with zeros (Use 10000 components so that is the number of walkers we are going to use). **Hint:** Use the function `np.zeros`
- Do a loop (1000 steps), and add a random normal number to each of the walkers
- Save, for each iteration, the standard deviation, you can use `np.var` and taking the square root.
- Plot the standard deviations (σ_x and σ_y) as a function of the step.
- A diffusion process is characterized because the standard deviation increase as the square root of the time, test this with a `curve_fit` such as

$$f(t) = a\sqrt{t} \tag{14}$$

  The value of a is related with the so called, Diffusion constant which describes the *speed* of the diffusion.

Entropy - Information Theory

On information theory, Shannon defines the entropy as

$$S = -\sum_i p_i \log(p_i) \tag{15}$$

in other words as the expected value of $\log(p)$.

Using that definition, one can calculate the entropy of different distributions, so we are going to compare different distributions. for each distribution, you have to

- Generate 1000000 points following the desire distribution.
- Calculate the histogram with 100 bins, you can do it with `matplotlib` such that `hist=plt.histogram(x,density=True,bins=100)`
- Calculate the entropy of the frequencies, on my previous line would be `hist[0]`.
  - You MUST normalize the histogram, so that take each frequency and divide it over the sum of all the frequencies. (So that the sum is 1).
  - As it is possible that you have some bins with 0 data there, put a condition that only calculate if `p[i]!=0` (The Logarithm diverges on 0).
- Save the result on a array-like.
- Repeat for at least 5 distributions, including
  - Uniform
  - Gaussian
  - Poisson

  Hint: Use the function `dir` (as it is on the slides to know how to implement other distributions).
- Plot the different entropies.
- Name the distribution with bigger entropy and discuss why is that particular one.

  Hint: Entropy means the lack of information.

This assignment finishes our course, take your time to do it well. I hope you learned, not just how to program, but to see problems from a different point of view.