Lab 9: Sled race!

SUNY Korea - Francois Rameau

Spring 2023

GitHub Classroom

Lab 9 – Gradient descent

Goals and preliminaries

Module

Creating and importing a module

Library

Installing a library using pip

OOP

Using OOP to build a small game

Gradient descent

A nice introduction to the concept of gradient descent

Installing Matplotlib

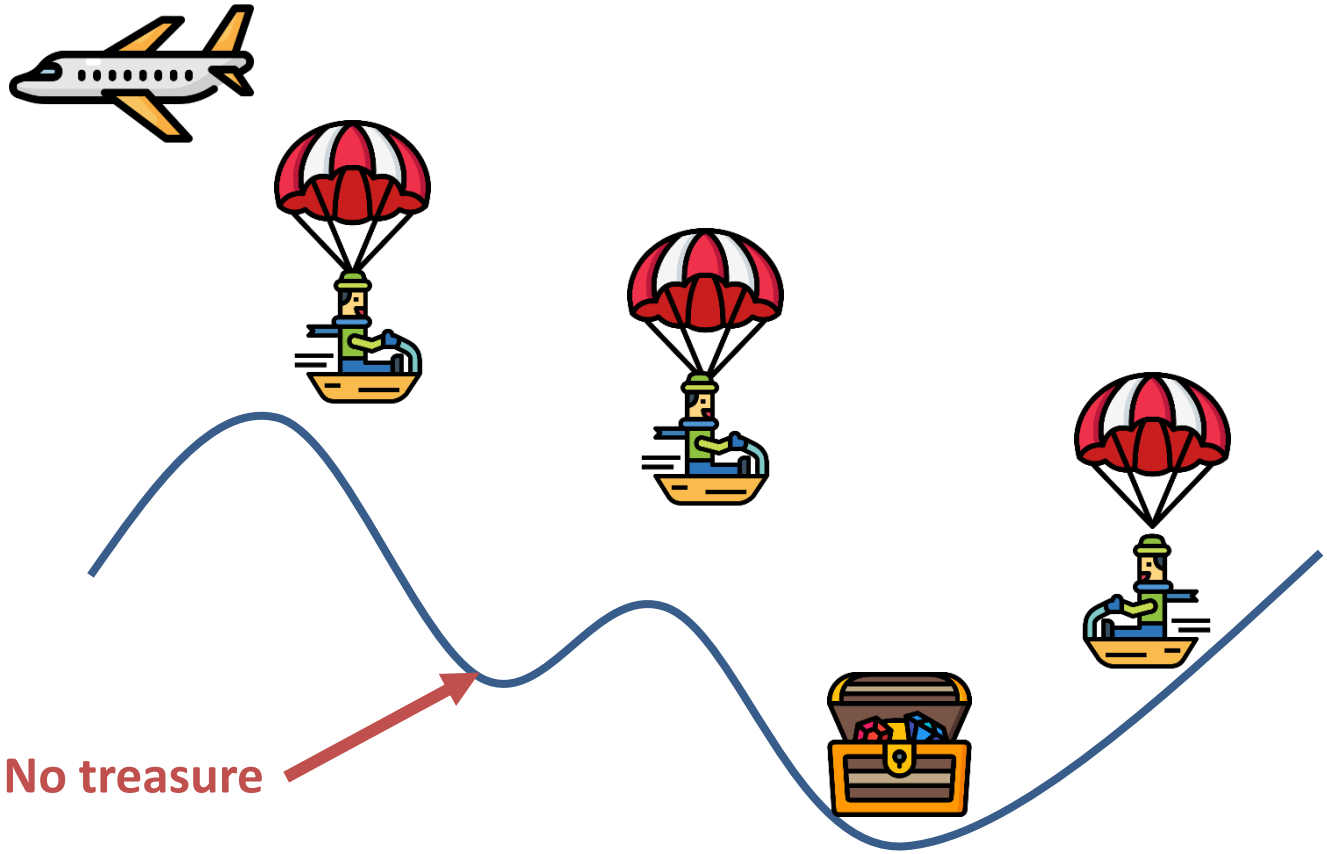
In this lab, we will plot a figure to illustrate the output of your code, this library is very commonly used in Python to plot curves: **Matplotlib**

You can install Matplotlib by typing the following in your terminal:

```
pip install matplotlib
```

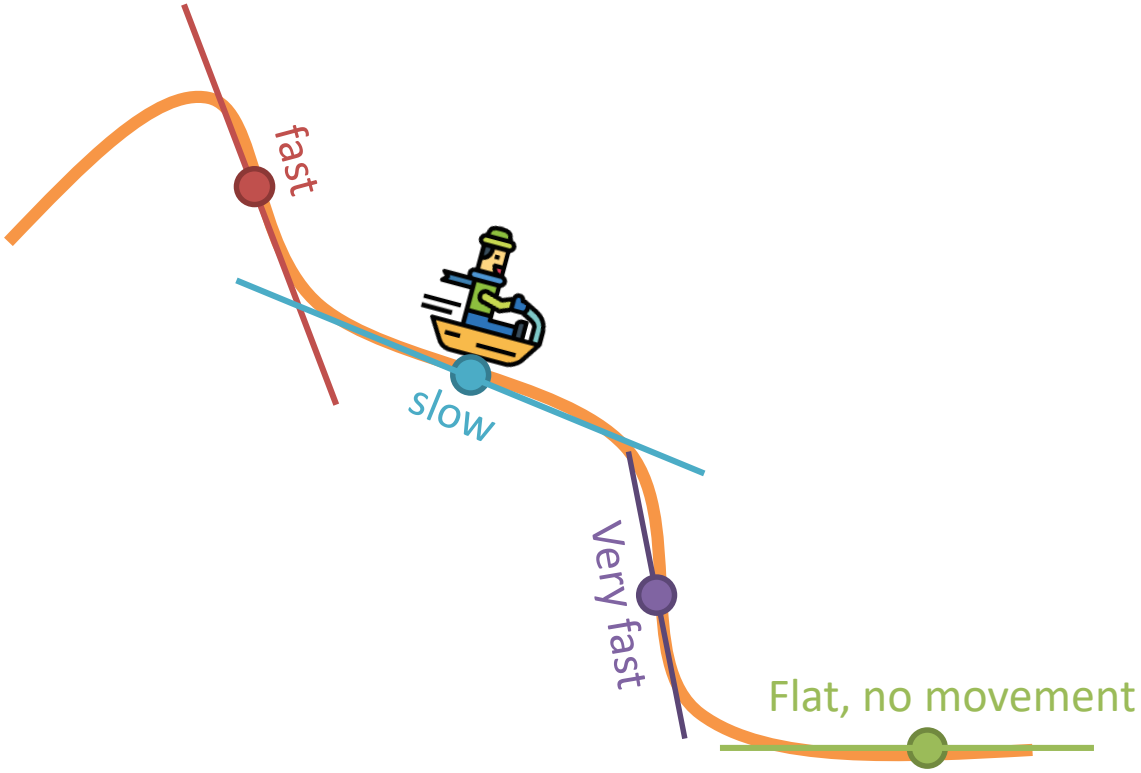
A little story

A treasure is at the **bottom** of a big mountain area. A race is starting between three people. They will land from parachutes near the treasure, with sleds attached to them. The first person to sled their way to the treasure gets to keep it.



A little story

To simulate how fast and in which direction our sleds will move, we need to determine the **slope (gradient)** of the mountain where the sleds are located.



I – The landspace

Equations of the landscape and slope

Landscape: Our terrain, is defined by the equation:

$$y = x^2 + 10\sin(x).$$

The sleds will randomly land within +/- 10 units from the treasure's location.

Slope's Role & Equation: The slope at a point influences the sled's speed and direction, with steeper slopes resulting in faster movement. The slope of our landscape at any position x is given by the derivative of our landscape equation:

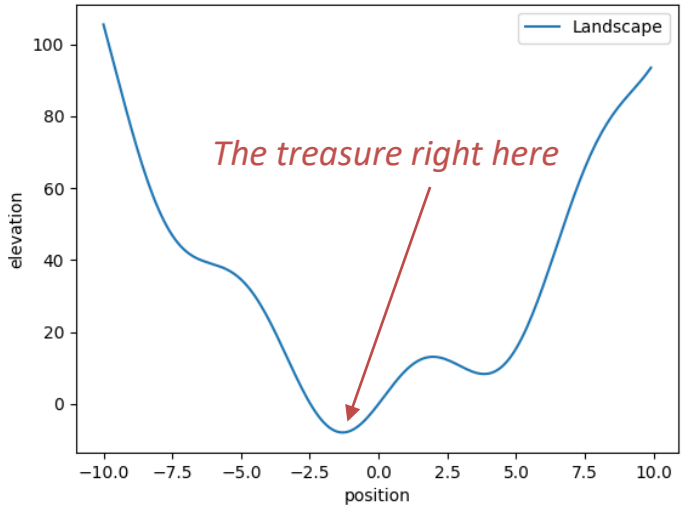
$$2x + 10\cos(x).$$

Your Task: Implement two Python functions in the module ***landspace.py***:

`elevation(x):` Returns the elevation at position x.

`slope(x):` Returns the slope at position x using the derivative equation.

IMPORT IT INTO YOUR MAIN CODE!



II – The sleds: attributes

You will create a class Sled and instantiate three competitors in the main

name: "Joe"
position: random uniform +/-10
speed: 0.05

name: "Jack"
position: random uniform +/-10
speed: 0.1

name: "John"
position: random uniform +/-10
speed: 0.08

Each sled has its own speed factor which will affect the convergence speed of the treasure (larger steps!).

Your Task: Implement the initialization function of the Sled class with the following characteristics: a name, a random initial position, and a constant speed.

III – The sleds: methods

Now that we have set our attributes, it is time to develop the two method of the sleds

1. move()

The move() method is what propels our sleds down the mountains. The new position of the sled can be calculated as follows:

$$new\_position = old\_position - speed * slope(old\_position)$$

Note the negative sign which specifies that we want to reach a minimum!

2. found_treasure()

This method checks if the sled is close enough to the treasure's location (the lowest point). If the sled's position is within a certain threshold of the treasure, it has found the treasure (return True, else return False)! This threshold is set in the main at 0.05

Your Task: Implement these two methods

IV – The race

We have our mountainous landscape and our three sleds - Joe, Jack, and John - ready to race! Now, it's time to see who will find the treasure first.

Rules of the race

Each sled will have maximum 60 turns to reach the treasure, if nobody finds the treasure before these sixty iterations, nobody is winning. In each turn, or minute, all sleds will 'move'. That is, they will adjust their position based on the landscape's slope at their current position. For each turn you will check if we have a winner!

Your Task: use your method move and break the loop whenever you have a winner

Conclusion

Today we have learned many things, but most importantly, you've implemented the concept of **gradient descent**, a crucial technique widely used for optimizing neural networks. Bravo!

If you have some time left:

- If I increase the speed of my sled, what is happening?
- Why are some sleds stuck in local minimums?
- Can you extend this problem in 3D?