Case pt. 2

RagnaRock Geo

March 28, 2019

Task 1

Ragnar is traveling with his girlfriend in the Caribbean Islands. He really likes islands, so he has placed a numerical value on every square kilometer on every island. The values range from 1 (Ragnar really does not like that square kilometer) to 10 (he really thinks that square kilometer is nice).

We can model this as a 2D numpy array of integers. The integers range from 1 to 10. The value 0 denote water. We define an island as a set of connected points surrounded by zeros. For this problem, we can have two kinds of connectivity[1]:

- A pixel (x, y) with value $v \neq 0$ is **4-connected** if at least one pixel in the set {(x-1, y), (x+1, y), (x, y-1), (x, y+1)} is also non-zero.
- A pixel (x, y) with value $v \neq 0$ is **8-connected** if at least one pixel in the set {(x-1, y-1), (x, y-1), (x+1, y-1), (x-1, y), (x+1, y), (x-1, y+1), (x, y+1), (x+1, y+1)} is also non-zero.

Given a 2D numpy array consisting of integer values , we want to find the island whose pixels have the greatest average value. I.e. let an island be defined as $\{x_1, \ldots, x_N\}$ of N connected points. Then the average value will be given by $\frac{1}{N}\sum_{i=1}^{N} x_i$. This is what we want to maximize.

In addition, Ragnar has Napoleon complex, so he consider only islands that are larger than m square kilometers.
Implement an algorithm that finds the island that Ragnar should bring his girlfriend with.
Input:

- np.array of shape (H, W) with $1 \leq H \leq 5000$ and $1 \leq W \leq 5000$. The values being between 0 and 10.
- a boolean indicator. If True, then use 4-neighbor connectivity. Otherwise use 8-neighbor connectivity.
- m, the minimum size of the island that Ragnar considers

[1] Strictly speaking, only 8-connectivity make sense for islands, but it's nice to support different kinds of connectivity :)

Task 2

Implement a feed-forward neural network, from scratch, using NumPy[1].

We have provided a skeleton class which describes the interface you will implement.
In addition, we highly encourage you to write some tests for your code, preferably using pytest [2].

Requirements

Programming

- The network should minimize the provided loss function (mean squared error [3])
- The network should support the following activation functions:
  - Sigmoid [4]
  - Relu [5]
- The network should calculate gradients correctly
- The output layer should not use an activation function
- Arbitrarily deep neural networks should be supported

Software Engineering

- The code should be readable
- The code should be modular; feel free to seperate your implementation into multiple files
- It should be easy to add new activation functions
- The code should not repeat itself (it should follow the DRY[6] principle)

References

[1] `http://www.numpy.org/`

[2] `https://docs.pytest.org/en/latest/`

[3] `https://en.wikipedia.org/wiki/Mean_squared_error`

[4] `https://en.wikipedia.org/wiki/Sigmoid_function`

[5] `https://en.wikipedia.org/wiki/Rectifier_(neural_networks)`

[6] `https://en.wikipedia.org/wiki/Don%27t_repeat_yourself`