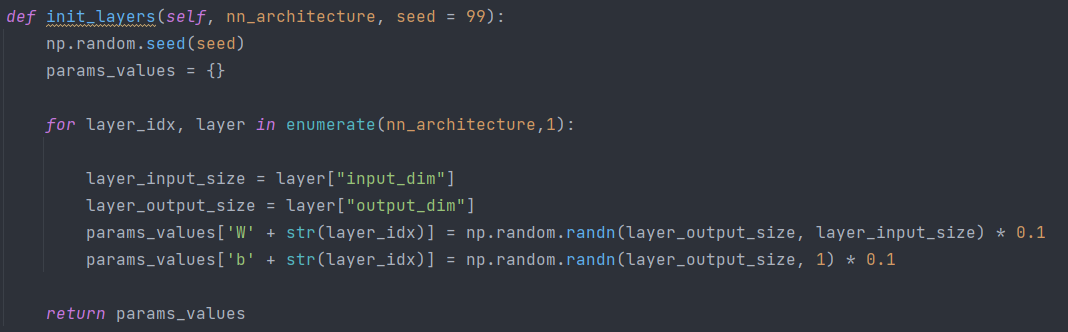
Layer init : 初始化參數W, b (W1代表第一層)



Forward : 分成single layer, full forward。

* single layer只做一層該做的事 : input \* W + b並回傳output(經 activate function), Z(未經activate function)。
* full forward 儲存single layer輸出，memory{}紀錄每一層output, Z 並回傳最後的output和memory 給後續計算使用。

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

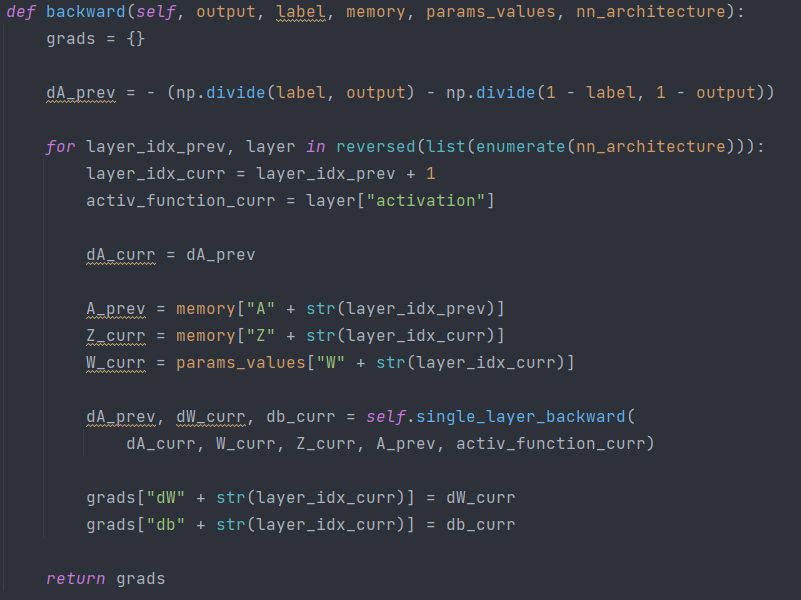
自動產生的描述

Backward: 分成single layer, full backward。

* single layer 只做一層該做的事 :
  1. 將self.error\*der\_sigmoid(output) 得到 dz\_dc
  2. 將dz\_dc @ pre\_a(前一層output) 得到 dW (Wwight的更新量)
  3. Sum(dz\_dc) 得到 db(bias的更新量)
  4. 將當前W@ dz\_dc得到前一層的dA(也就是前一層的loss數值)
  5. 最後回傳2, 3, 4
* Full backward : 儲存single layer輸出裡的dW, db，用於接下來update使用

一張含有 文字 的圖片

自動產生的描述



Update :

New\_W = (old W - dW) \* lr

New\_Bias = (old B - dB) \*lr

一張含有 文字 的圖片

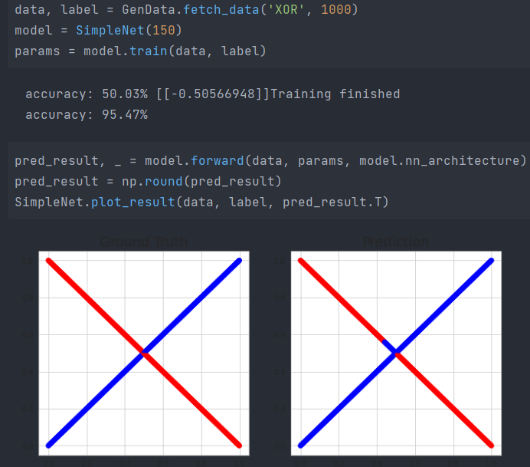
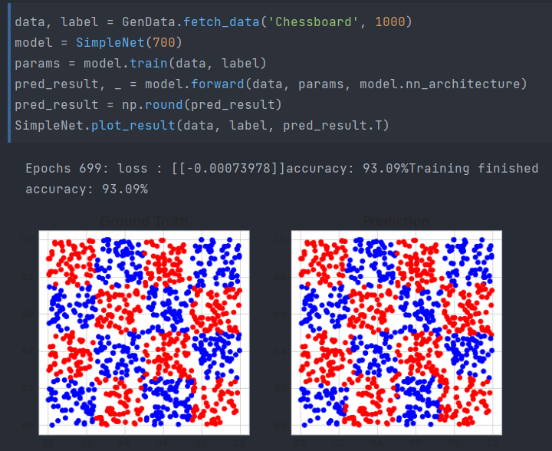
自動產生的描述

除了sigmoid 我在最後一層前都使用relu，並在backward用der\_relu:

一張含有 文字 的圖片

自動產生的描述

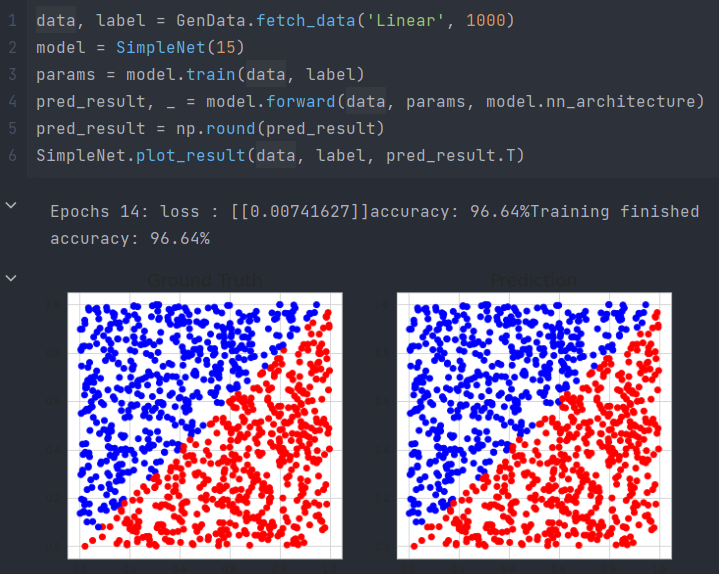
Finally output : 我’linear’, ‘XOR’都只分別讓他們train 100,150 epochs 就達到99%, 95%



ablation experiment : use relu vs only use sigmoid

使用了relu後，在’linear’的case裡train 15個epochs，acc就到達96.64，

而只用sigmoid必須train 200 epochs，acc才能到達96.19%

use relu only use sigmoid