深度學習

Lab1 : back-propagation

312605003

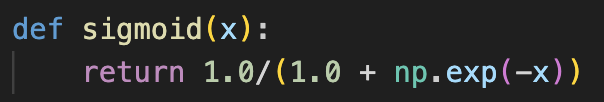
王語 2023/7/29

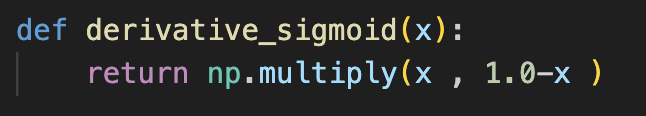
1. Introduction

本次實驗在僅使用標準函式庫以及Numpy的前提下實作神經網路，此神經網路可以使用不同的激勵函數(Active Function)、不同的優化器(Optimizer)以及兩個可調整神經元數目的隱藏層，以上參數可以透過 Command Line Arguments來設定，具體操作方式詳見README.md。

實驗中使用正向傳播(Forward propagation)計算預測值，透過MSE計算Loss，使用反向傳播(Back-propagation)計算梯度，透過優化器更新參數，並將結果輸出於output.txt。

1. Experiment setups
   1. Sigmoid functions

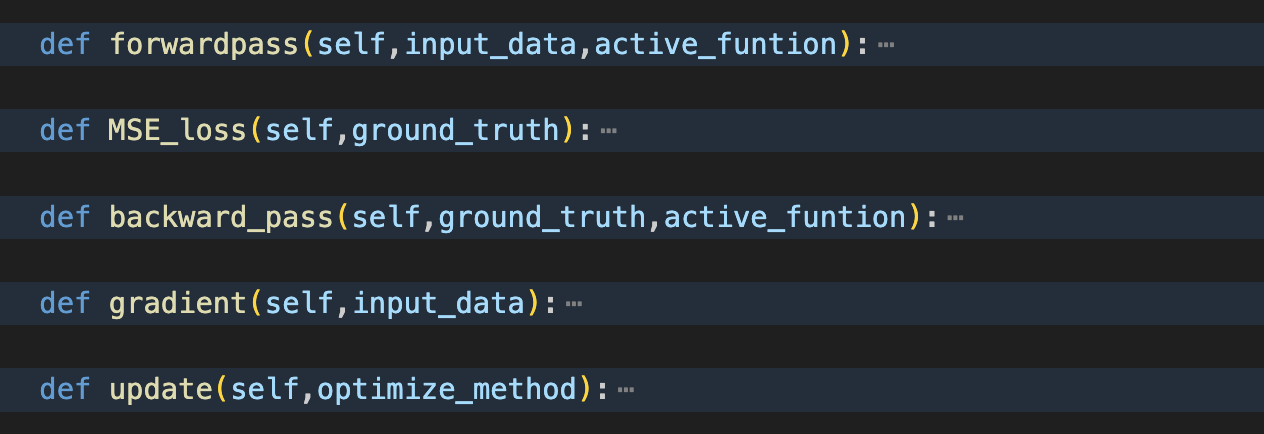
用於正向傳播中的sigmoid function:

用於反向傳播中的derivative sigmoid function:

* 1. Neural network

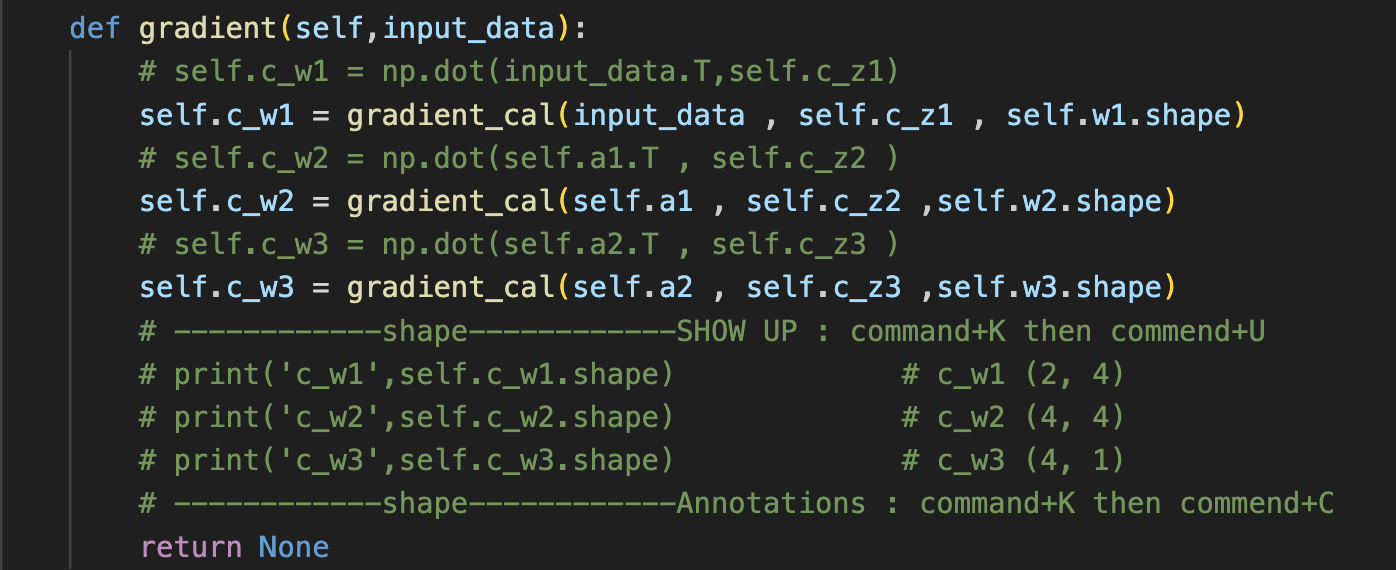
通過 two\_layer\_network 這個class來建立實驗需要的Neural network，在\_\_init\_\_中會根據訓練參數的設定以及神經網路參數的設定，建構神經網路並且生成初始參數。

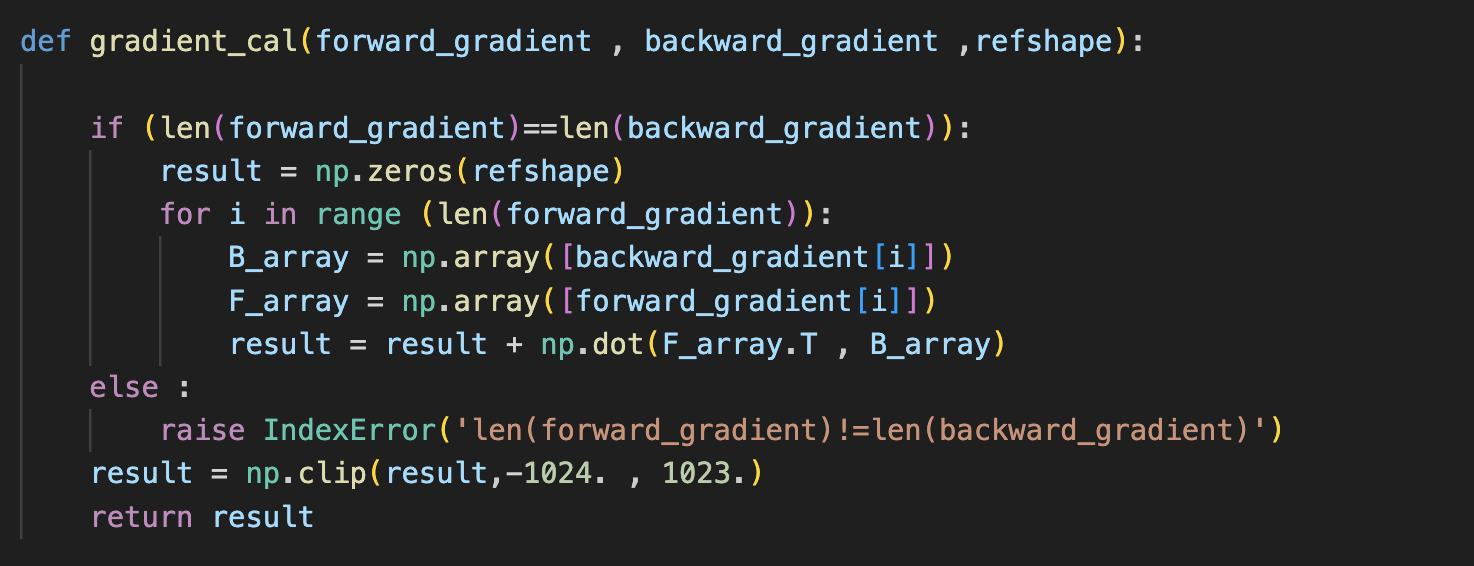


 此外是two\_layer\_network的方法說明，這個class中包含以下五個功能，正向傳播、Loss計算、反向傳播、梯度計算、參數更新。

* 1. Backpropagation

即根據激勵函數的設定，將loss由output向input傳遞，計算出Backward pass。

 再將Forward Pass 與 Backward Pass 相乘即可得到梯度。



1. Results of your testing
   1. Screenshot and comparison figure

|  |  |
| --- | --- |
| Linear data | XOR data |
|  |  |

結果完全吻合，準確率100%

* 1. Show the accuracy of your prediction

|  |  |
| --- | --- |
| Linear data | Acc = 1.0 |
|  | |
| XOR data | Acc = 1.0 |
|  | |

* 1. Learning curve (loss, epoch curve)

|  |  |
| --- | --- |
| Learning rate = 0.1 | |
| Linear data | XOR data |
|  |  |
|  |  |

* 1. Anything you want to present

|  |  |
| --- | --- |
| output | |
| Linear data | XOR data |
|  |  |

記錄該次訓練設定。

此處的參數是程式預設數值，在Discussion章節中的對照組皆使用預設數值。

1. Discussion
   1. Try different learning rates

|  |  |
| --- | --- |
| Learning rate = 0.1 (對照組) | |
| Linear data | XOR data |
|  |  |
| 20000 epoch : loss = 0.007732903474659361 | 20000 epoch : loss = 0.030861509686194014 |
| Acc = 1.0 | Acc = 1.0 |
| Learning rate = 0.01 （實驗組） | |
| Linear data | XOR data |
|  |  |
| 20000 epoch : loss = 0.10014453046463924 | 20000 epoch : loss = 0.24647790519187202 |
| Acc = 0.94 | Acc = 0.52 |

對比Learning rate = 0.1 與 Learning rate = 0.01的結果可以發現 Learning rate = 0.01 太小了，導致loss還沒有收斂到最小值，要增加訓練次數才能得到更好的結果。其中XOR data的loss最終落在0.246，若所有數據都由隨機的方式猜測loss將會貼近0.25，說明了這個神經網路並沒有從訓練資料中得到優化，在測試資料中的準確率也只有0.52，代表訓練非常不良，應調整訓練參數。

|  |  |
| --- | --- |
| Learning rate = 1 （實驗組） | |
| Linear data | XOR data |
|  |  |
| 20000 epoch : loss = 0.0004890141339678335 | 20000 epoch : loss = 9.617784260566798e-05 |
| Acc = 0.99 | Acc = 1.0 |

兩個資料源在Learning rate = 1都取得了不錯的表現，比起Learning rate = 0.1，loss也收斂到更小的數值，但是在Linear data中的準確率卻沒有比較好，推測是出現了Over fitting的現象，應該減少訓練次數。

|  |  |
| --- | --- |
| Learning rate = 1 （驗證組） | |
| Linear data | XOR data |
|  |  |
| 15000 epoch : loss = 0.0007520826722752352 | 15000 epoch : loss = 0.000147078690746444 |
| Acc = 1.0 | Acc = 1.0 |

將訓練次數下降至15000次變得到了1.0的準確率，說明Learning rate = 1更加有效率。

* 1. Try different numbers of hidden units

|  |  |
| --- | --- |
| Units = 4 (對照組) | |
| Linear data | XOR data |
|  |  |
| 20000 epoch : loss = 0.007732903474659361 | 20000 epoch : loss = 0.030861509686194014 |
| Acc = 1.0 | Acc = 1.0 |

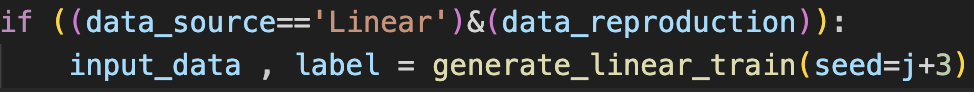
|  |  |
| --- | --- |
| Units = 2 (實驗組) | |
| Linear data | XOR data |
|  |  |
| 20000 epoch : loss = 0.008657442206315567 | 20000 epoch : loss = 0.24843621581680947 |
| Acc = 0.99 | Acc = 0.52 |

下降成2 units在XOR data上會失去精準度，表示模型太過簡易，無法達成任務要求。

|  |  |
| --- | --- |
| Units = 8 (實驗組) | |
| Linear data | XOR data |
|  |  |
| 20000 epoch : loss = 0.0071275703335514275 | 20000 epoch : loss = 0.01064488716425127 |
| Acc = 0.98 | Acc = 1.0 |

將units提升一倍後在Linear data上並沒有得到更好的表現，反而下降的2個百分比，推測是因為訓練資料沒有隨著模型擴增而增加。

|  |  |
| --- | --- |
| Units = 8 (驗證組) | |
| Linear data | Linear data |
|  |  |
| 20000 epoch : loss = 0.010447685355981123 | 20000 epoch : loss = 0.01057658215122545 |
| 每500 epoch生成新的訓練資料 | 每5000 epoch生成新的訓練資料 |
| Acc = 0.99 | Acc = 1.0 |



通過在訓練過程中加入新的訓練資料，可以改善這個問題，雖然會讓學習曲線出現震盪，但是最終獲得較好的準確率。

* 1. Try without activation functions

|  |  |  |
| --- | --- | --- |
| Sigmoid activation function (對照組) | | |
| Linear data | XOR data | |
|  |  | |
| 20000 epoch : loss = 0.007732903474659361 | 20000 epoch : loss = 0.030861509686194014 | |
| Acc = 1.0 | Acc = 1.0 | |
| Without activation function (實驗組) | | | |
| Linear data | | XOR data | |
|  | |  | |
| 20000 epoch : loss = 0.13689657367274508 | | 200000 epoch : loss = 0.2887139107611548 | |
| Acc = 0.86 | | Acc = 0.333 | |

不使用激勵函數梯度絕對值會快速下降(下降幅度應該更大，但是我為了防止梯度過大因此加入了上下限，功能類似 torch.clamp( ) )，導致無法更新參數。

1. Extra
   1. Implement different optimizers.

|  |  |
| --- | --- |
| gradient descent with momentum | |
| Linear data | XOR data |
|  |  |
| 200000 epoch : loss = 0.004492213994639331 | 200000 epoch : loss = 0.003352094072948981 |
| Acc = 1.0 | Acc = 1.0 |

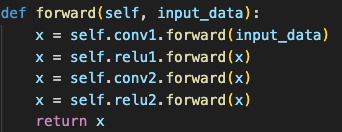
* 1. Implement different activation functions.

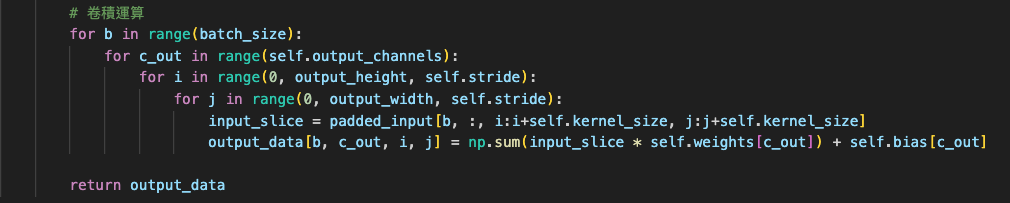
|  |  |
| --- | --- |
| tanh activation function | |
| Linear data | XOR data |
|  |  |
| 200000 epoch : loss = 0.0018610136203773735 | 200000 epoch : loss = 7.47451314505538e-05 |
| Acc = 0.99 | Acc = 1.0 |

* 1. Implement convolutional layers.

我有實現簡易的卷積神經網路以及相關計算，但是我認為卷積層並不適合用於眼前的任務，單一一筆Input Data 是shape為 (1,2) 的陣列，在不使用Padding的前提下要對這樣的陣列進行卷積，卷積核的大小只能設定為 （1,2）或是 （1,1），但是這樣的運算並不能體現出「特徵提取」的精髓，因為在數學上，只是將原本的矩陣乘上一組係數而已，因此我另外定義了輸入資料。

卷積神經網路最長被使用在圖像辨識，因此我假設輸入資料是10張28x28的RGB圖像。

此卷積神經網路共有兩層卷積層以及活化層

通過卷積運算得到預測結果