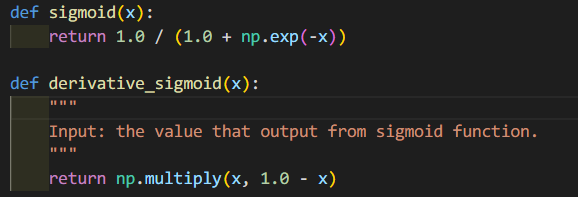
**DLP LAB1**

1. **Introduction**

實作具有2層hidden layer的Fully-Connected Neural Network來將預測input data的分類，並藉由Backpropagation來加速Gradient Descent中計算Gradient步驟，其中實作可以任意修改層數功能。

1. **Experiment setups**
2. Sigmoid functions

為非線性方程式，利用它作為activation function解決非線性問題。



上圖為sigmoid以及derivative sigmoid function，sigmoid function被用於Forward Pass的計算中，derivative sigmoid function則用於Backward Pass的計算，其中需特別注意的是，derivative sigmoid function的input為經過sigmoid function的value。

1. Neural network



2層Hidden layer內皆有10個hidden units，Learning rate設為0.1，Loss function使用Cross Entropy，

一張含有 文字 的圖片

自動產生的描述

當Loss小於0.001時停止，Epoch上限為100000。

1. Backpropagation
2. 初始化Neural Network的所有Weight
3. 由input layer往output layer做forward pass，計算出所有neuron的output
4. 再由Neural Network的output與實際label計算出loss (誤差)
5. loss由output layer 往input layer做backward pass (相當於一個反向的Neural Network)，並計算出每個weight對loss的偏微 (即該neuron對誤差的影響)
6. 利用weight對loss的偏微去更新weight
7. 重複步驟ii.~v.直到loss夠小
8. **Results of your testing**
9. Screenshot and comparison figure

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

可以看到兩種data的預測皆是準確的。

1. Show the accuracy of your prediction

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

兩種data的準確率皆為100%

1. Learning curve

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

由上表可以看到Linear data較快收斂，而XOR data則是先維持在差不多的loss之後，才逐漸收斂。

1. Anything you want to present

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

由上表可以看到Network對於兩種data的輸出值，Linear data的輸出值都非常接近0或是1，而XOR data的輸出值相對來說比較沒那麼接近0或1。

1. **Discussion**
2. Try different learning rates

固定兩層Hidden unit皆為10。

1. Linear

|  |  |  |
| --- | --- | --- |
| 0.1 | 1 | 10 |
|  |  |  |
|  |  |  |
|  |  |  |

由上表可以看到雖然結果都是預測正確，但是在learning curve的部分是有些差異，設為1時比較快達到較小的loss值，設為10時則更快收斂，且有時loss會往上升，因此Learning curve看起來有上下震盪的感覺。

1. XOR

|  |  |  |
| --- | --- | --- |
| 0.1 | 1 | 10 |
|  |  |  |
|  |  |  |
|  |  |  |

同樣結果也是都預測正確，但是相較於linear data，更明顯看到設為1很快收斂，而設為10則又更快收斂，並且過程中loss有大幅度的上升，因此learning curve看起來震盪很大。

1. Try different numbers of hidden units

固定learning rate為0.1。

1. Linear

|  |  |  |
| --- | --- | --- |
| (2, 2) | (10, 10) | (100, 100) |
|  |  |  |
|  |  |  |
|  |  |  |

結果都預測正確，設為(2, 2)時初始的Loss最小，再來是(10, 10)，而(100, 100)則是最大的，不過最後都是收斂到差不多的loss。

1. XOR

|  |  |  |
| --- | --- | --- |
| (2, 2) | (10, 10) | (100, 100) |
|  |  |  |
|  |  |  |
|  |  |  |

設為(2, 2)的結果讓準確率下降了約24%，loss下降較為緩慢，並且數值也較高，

而設為(100, 100)時的learning curve更快達到收斂，並且不像(10, 10)有先維持一下才往下降。

1. Try without activation functions

固定兩層Hidden unit皆為10，learning rate為0.1，without activation function只有在output layer使用sigmoid function。

1. Linear

|  |  |
| --- | --- |
| With activation function | Without activation function |
|  |  |
|  |  |
|  |  |

除了剛開始Loss值較大外，沒有activation function對於Linear data似乎是沒有甚麼很大的影響。

1. XOR

|  |  |
| --- | --- |
| With activation function | Without activation function |
|  |  |
|  |  |
|  |  |

不同於Linear data，沒有activation function對於XOR data則是有非常大的影響，準確率直接下降了約53%，並且在很前面的epoch開始loss就一直維持在很大的值，不再繼續下降。

1. Anything you want to share

修改learning rate後，計算Loss function時會overflow，之後將Network加一個很小的數值丟進去計算就解決了。

1. **Extra**
2. Implement different optimizers
3. Implement different activation functions
4. Implement convolutional layers