Safiya Rehmat 18200494

Connectionist Computing Programming Assignment

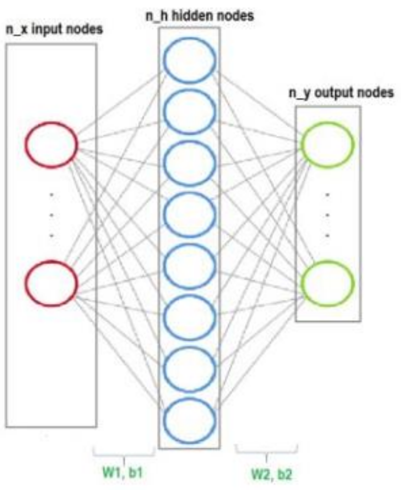
Inroduction

This notebook implements networks to perform 3 different tasks:

```
1. Train a network to learn XOR
2. Train a network to learn sin(x1-x2+x3-x4) for inputs x1,x2,x3 and x4
3. Train a network to perform letter recognition
```

Network Architecture

The figure below shows the architecture of networks used in this assignment:



n_x and n_y are determined by the shape of the input and target outputs. n_h is a hyperparameter, with default value 3.

For tasks 1 and 2, we have only 1 output unit as they are regression tasks. Task 3 is a multi-class classification and thus has more than 1 output units.

Forward Propogation

- In case of regression problems, we use tanh units for both hidden and output layers. The loss is calculated using mean squared error formula.
- In case of multi-class classification, we use tanh for hidden units and softmax for output layer. The loss is computed as the cross entropy loss.

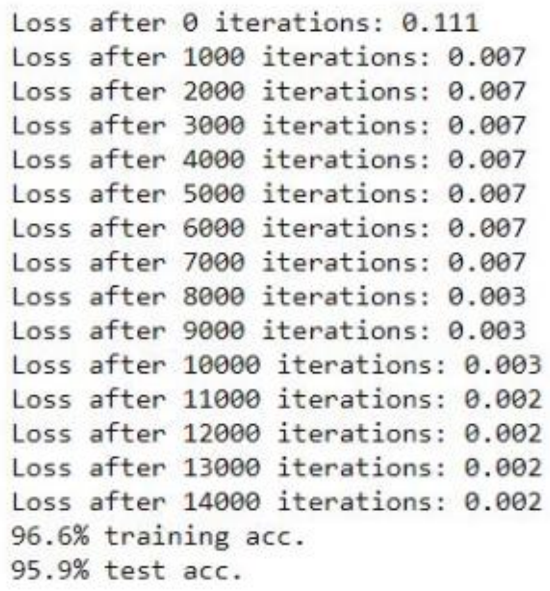
Results

The performance of the model presented in this assignment is as follows for each task:

Task 1 : Learning the XOR function

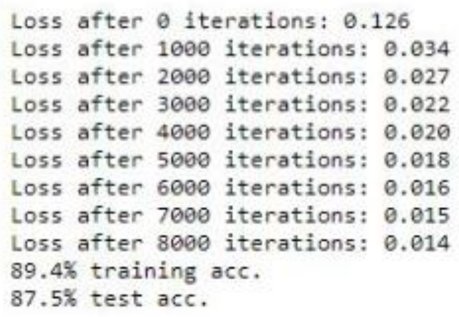
```
Loss after 0 iterations: 0.174
Loss after 1000 iterations: 0.000
Loss after 2000 iterations: 0.000
99.6% training acc.
99.6% test acc.
```

Task 2 : Learning the Sine function



```
Loss after 0 iterations: 0.111
Loss after 1000 iterations: 0.007
Loss after 2000 iterations: 0.007
Loss after 3000 iterations: 0.007
Loss after 4000 iterations: 0.007
Loss after 5000 iterations: 0.007
Loss after 6000 iterations: 0.007
Loss after 7000 iterations: 0.007
Loss after 8000 iterations: 0.003
Loss after 9000 iterations: 0.003
Loss after 10000 iterations: 0.003
Loss after 11000 iterations: 0.002
Loss after 12000 iterations: 0.002
Loss after 13000 iterations: 0.002
Loss after 14000 iterations: 0.002
96.6% training acc.
95.9% test acc.
```

Task 3 : Learning Letter Recognition



```
Loss after 0 iterations: 0.126
Loss after 1000 iterations: 0.034
Loss after 2000 iterations: 0.027
Loss after 3000 iterations: 0.022
Loss after 4000 iterations: 0.020
Loss after 5000 iterations: 0.018
Loss after 6000 iterations: 0.016
Loss after 7000 iterations: 0.015
Loss after 8000 iterations: 0.014
89.4% training acc.
87.5% test acc.
```

Conclusion

- We get 95% or above accuracy in the first 2 tasks on the test set, the model has learned these tasks pretty well.
- For task 3, we get 87.5% accuracy, which is pretty decent. We may be able to optimize it further by experimenting with the hyper-parameters