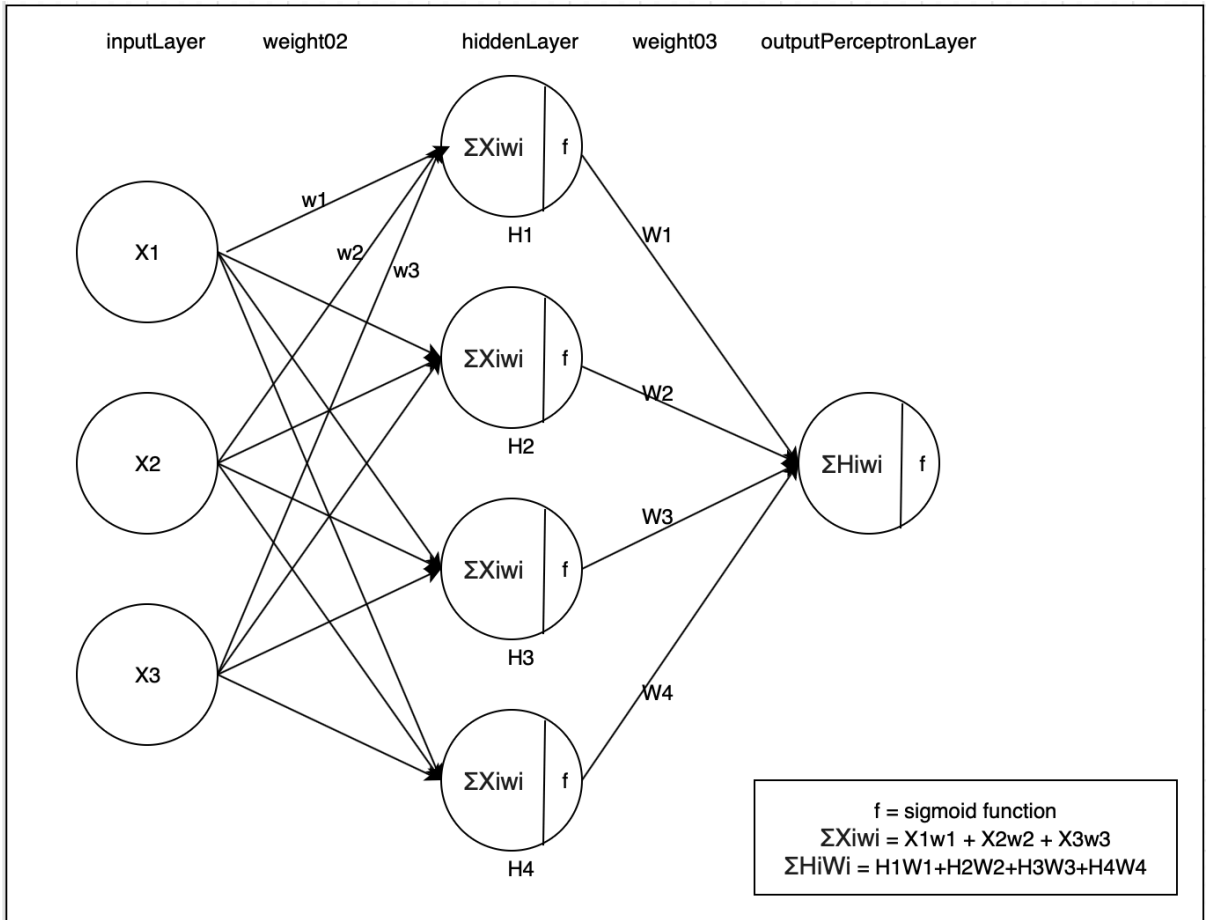
Q1. Draw a multilayer neural network as a diagram as practiced during last lecture. The naming convention must follow the one used in the Python code. The final diagram should only include variable and function names. The diagram must be drawn with a computer.



Q2. How does the multilayer neural network compare to the single layer neural network? Take two problems (two different referenceOutput), one solvable by both and on only by multilayer neural network. Get the values of the mean output error np.mean(np.abs(outputError)) for 900 iterations (both problems and both neural networks). Plot all four variables as a graph and include it in your report.

Comparing to the single layer neural network, multilayer neural network works more complexly. In the code, the multilayer neural network is functionalized as a one hidden layer and 4 node (dimension). Unlike the single layer neural network, multilayer neural network can be more classified. Therefore, if not overfed, using multilayer can make the neural network able to learn by further subdivisions. Taking the number recognition program as an example is easy to understand. A hidden layer can be functioned as recognizing the stroke of each part of a number. On the other hand, if there is too much layer or too much nodes in the hidden layer. It is also called overfeeding. In this situation, neural network might not learn well, moreover, the time and cost of the program might be inefficient.

For the code, there is two referenceOutput, [[0], [1], [1], [0]] and [[0], [0], [1], [1]]. For [[0], [0], [1], [1]], both the single and multilayer works without error. However, for [[0], [1], [1], [0]], it does not navigate to it, and for seed(1) it converges to [[0.5], [0.5], [0.5], [0.5]]. However, if a hidden Layer is added, it is successfully navigated to the referenceOutput.
For this phenomenon, [[0], [1], [1], [0]] gives error, because the neural network is not complex enough. However, by adding layers it allows the network to be complexed enough to enable the learning process.
Based on the four variable the following is the graph.
The blue graph is the SingleLayer with the referenceOutput1 which is [[0],[1],[1],[0]].
As seen in the graph, the outputErrormean converges to 0.5 instead of 0.
The orange graph is the MultiLayer(one hidden layer) with the referenceOutput1.
Unlike the blue graph it converges to 0.0 if iteration goes on.
The green graph is the SingleLayer with the referenceOutput2 which is [[0],[0],[1],[1]].
The red graph is the MultiLayer with the referenceOutput2.
For green and red graph, green graph seems to navigate to 0.0 faster at beginning. However, in the end red graph converges to 0.0 at a higher speed.