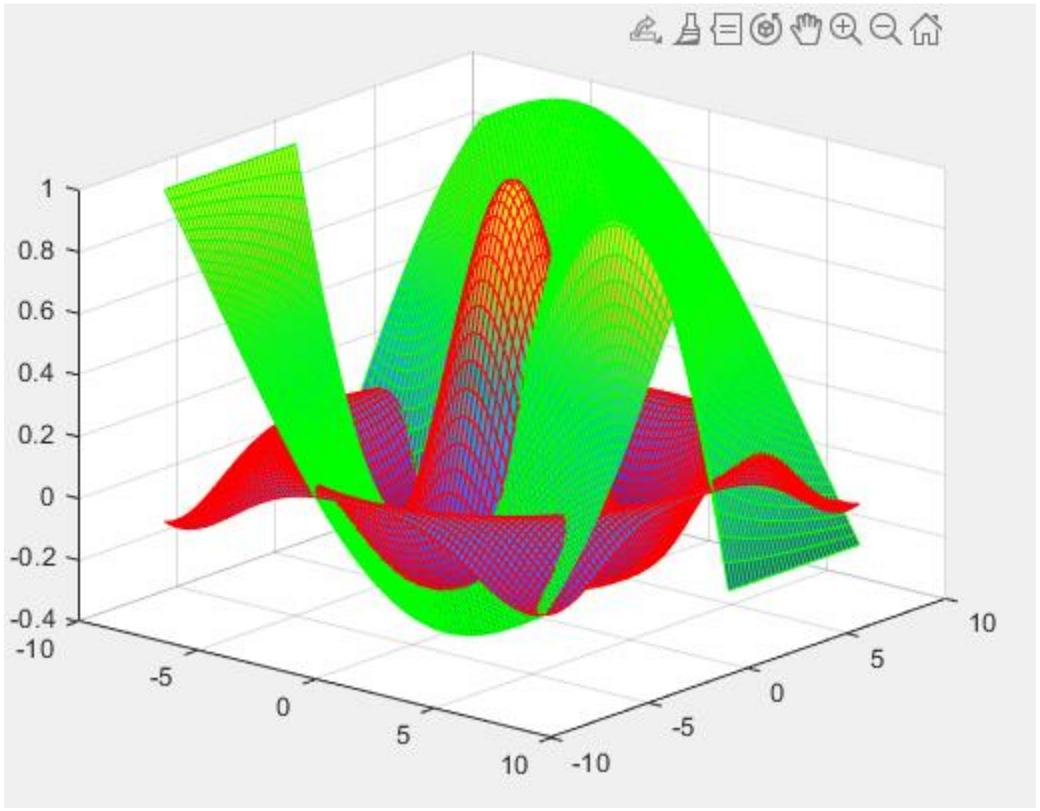
# Machine Learning – MTRE 4490-01

Lab 3: MATLAB Neural Network

Pierre Coiron



Class Instructor: Dr. Ying Wang

Kennesaw State University

Southern Polytechnic College of Engineering

# Table of contents

## Contents

## Abstract

A neural network was created from scratch using MATLAB for a 3-dimensional function. Although an accurate model was never produced, a proof of concept can be seen from the architecture of the network itself.

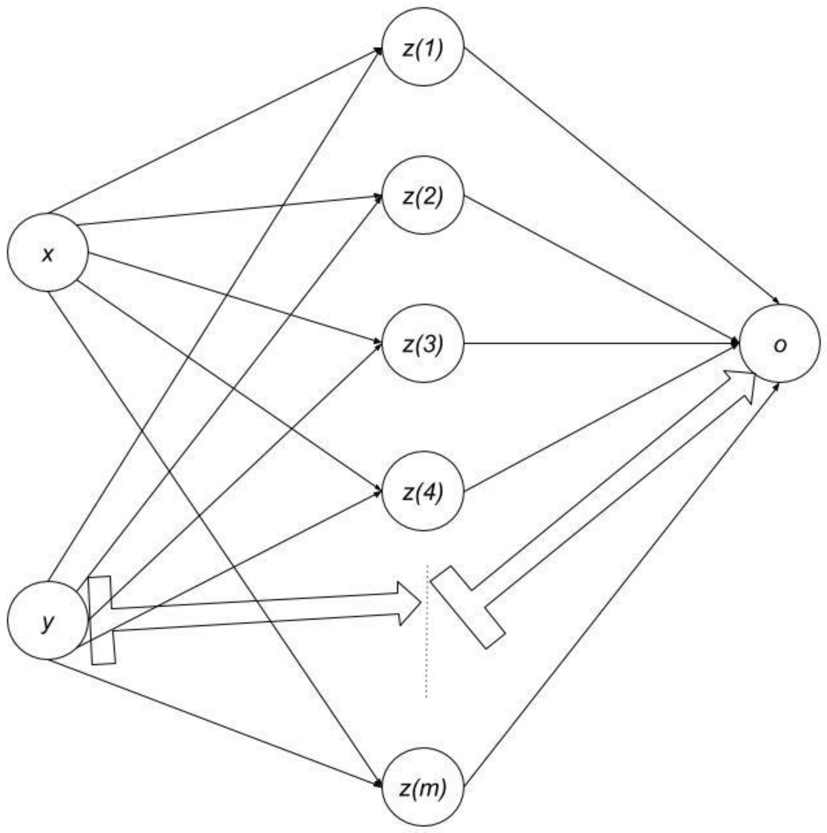
## Introduction

Single Layer Neural Networks can be very robust. In this lab, the attempt was made to make a network that takes in data from the solved equation mentioned in the appendix and predict an output based upon the training data.

The program was to also sample two weights and show the progression of how the selected values change over iterations of training.

## Network Architecture Explanation

Below is the visual representation of the architecture of the Neural Network:



The two cells on the left side, labeled “x” and “y” are the two input nodes. All input data is normalized and then sent through the input as a 2x1 matrix for every iteration. Both of the input layers have arrows pointing to every node in the middle layer, or “hidden layer”, denoted as “z”. The lines leading *into* the hidden layer are known as the “hidden layer input weights”. Such weights start off random but get better over iterations of training. Note that the hidden layer input weights is often denoted as *w*.

The middle layer is known as the hidden layer. The number of nodes within the hidden layer is one of the three meta-variables and can be changed easily as their declaration can be seen at the very top of the code for ease-of access. The middle layer is the applied sigmoid function to the sum of the products of the input later in the weight connecting the input layer to the specific hidden layer node. The formula for a hidden layer node can be seen below:

$$n(4) = \frac{1}{1 + e^{-((x \times w_x) + (y \times w_y))}}$$

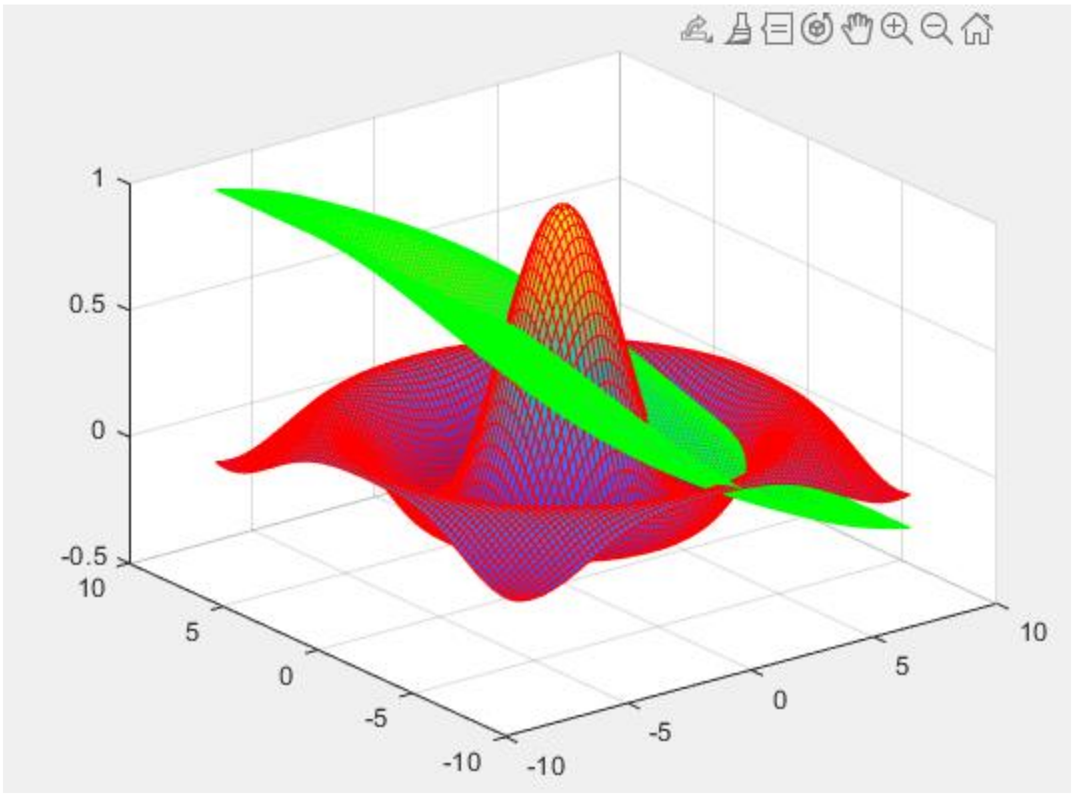
The lines leading from the hidden layer to the output is known as the “Hidden Output Weights”. Just like the hidden layer input weights, they are randomly generated at first, but get better over time. The values for the hidden output weights can be denoted as *v*.

Finally, the single output node on the right side is the sum of the products of each hidden layer node with their corresponding hidden layer output weight.

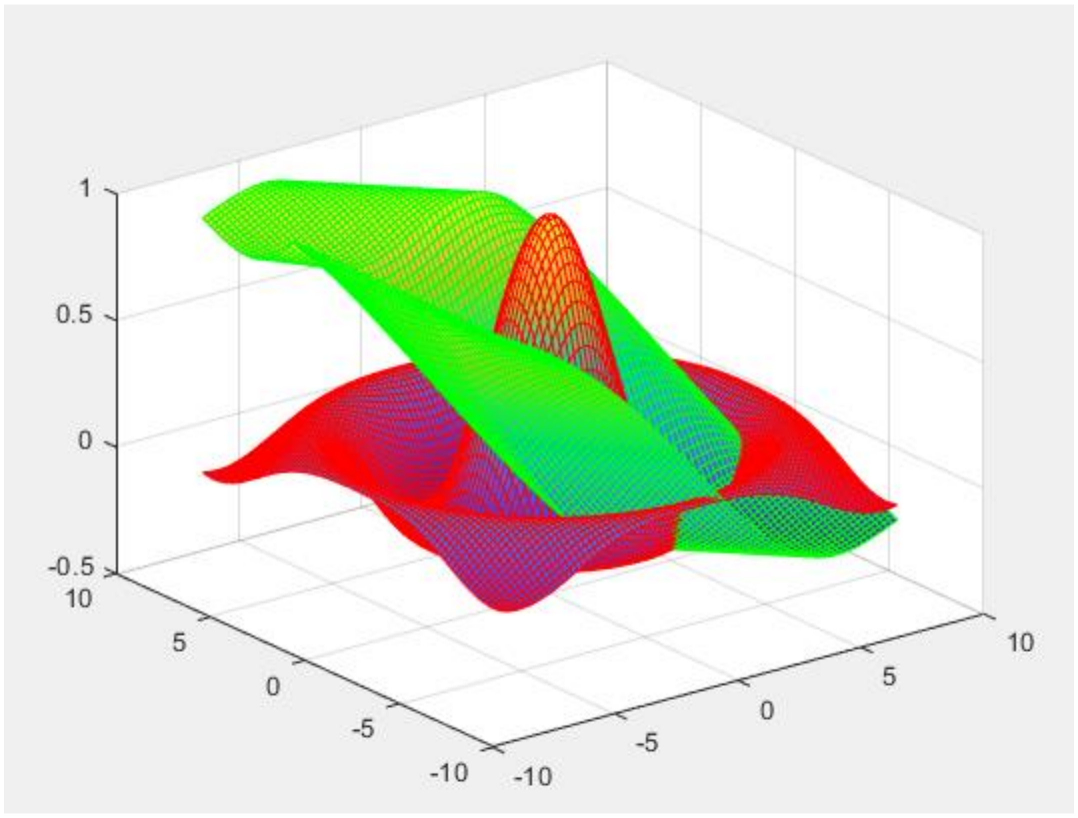
## Results

As mentioned before, unfortunately the results were disappointing and it was difficult to find an accurate model given the time spend tuning the meta-variables. Some results can be seen below along with their respective set of meta-variables:

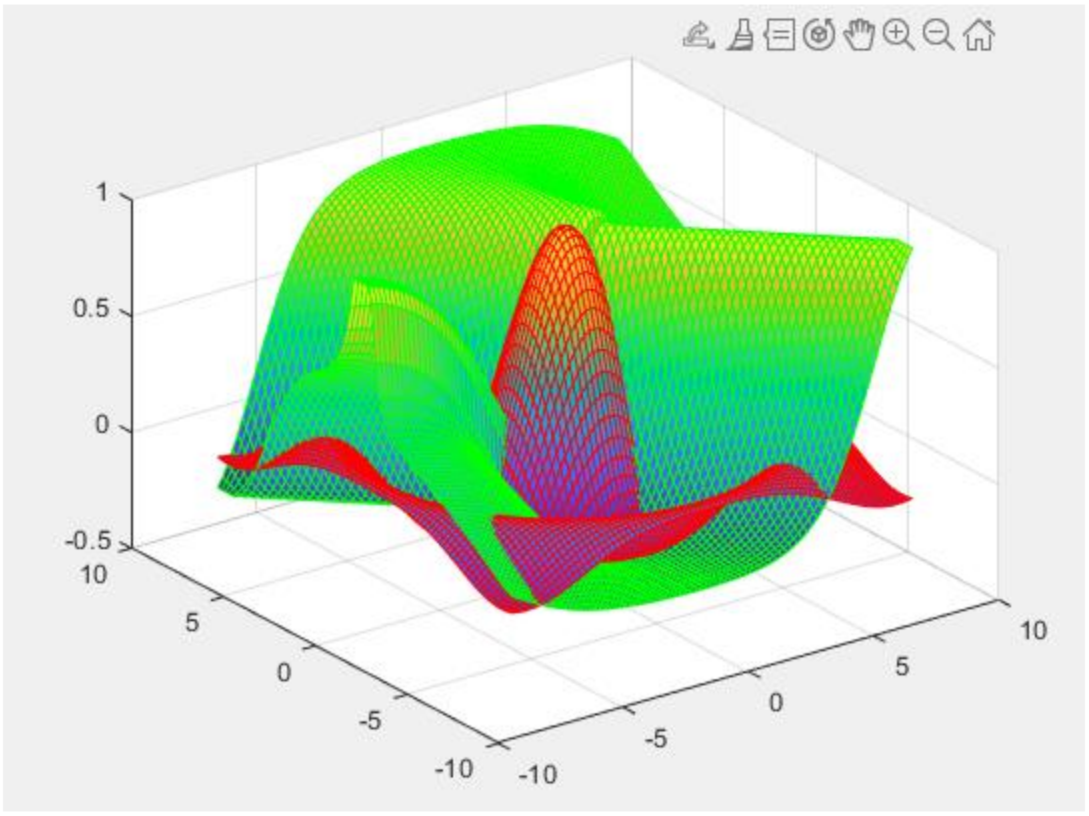
| Number of Hidden Layers: 50 | $\zeta = 0.005$ | Number of Training Steps: 25000 |
|---|---|---|



| | | |
|---|---|---|
| Number of Hidden Layers: 200 | $\zeta = 0.005$ | Number of Training Steps: 35000 |



| Number of Hidden Layers: 200 | $\zeta = 0.01$ | Number of Training Steps: 35000 |
|---|---|---|



## Conclusions

Of course, one of the obvious methods of improving the outcome of the Network is to continue "tuning" the network's "meta-variables". The $\zeta$ value could also be made dynamic. Also, by using the error function, I meta-network could be made to find the values of the meta-variables that would best fit the Neural Network at-hand.

## Appendix

Predicted Function:

$$z = \frac{\sin\sqrt{x^2 + y^2}}{\sqrt{x^2 + y^2}}$$