Raiid Ahmed Hw 6 and 7

Project 6.1:

Given!

Back propagation

1 hidden layer

bias on hidden / outputs

Bipolar sigmoid

$W = [-.5, .5] = V$

$\alpha = .05, .25, .5$

1000, 10000, 25000 epochs/alpha

2 inputs, 2 hidden units, 1 output

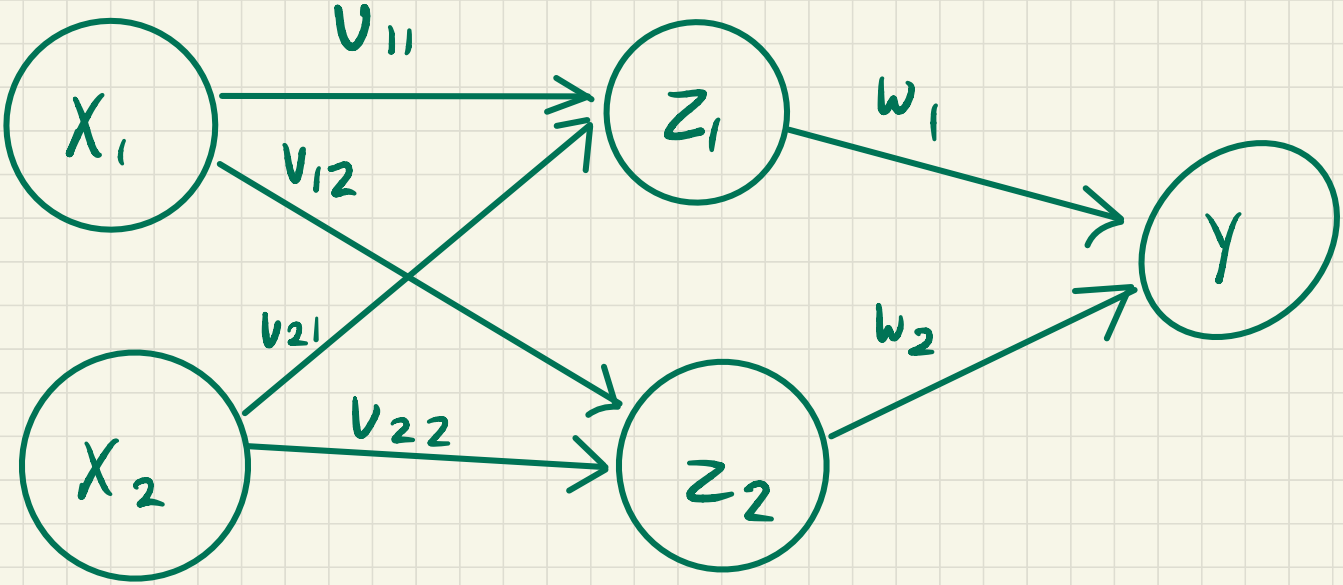
Test Data:

$$s = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ -1 & -1 \end{bmatrix} \quad t = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

Find!

- initial / final weights
- network response

Net Design:



Net Construction!

Back propagation net, On-line learning

2 unit inputs, 1 unit output

update weights/bias w/ generalized delta rule

V will be 2x2 matrix, W will be 2x1

Stopping condition will be at 1000, 10000, 25000 epochs

Activation function will be bipolar sigmoid

Algorithm:

- Define $f(x)$ as bipolar sigmoid, $g(x)$ as bipolar step
- Allocate test sets, weights, and variables
- Initialize w and V w/ random values
- Select α, randomly select a training vector
- Train feed forward
  - Calculate z values

$$z\_in_j = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

$$z_j = f(z\_in_j)$$

- Calculate Y values

$$y_{in} = w_0 + \sum_{j=1}^{p} z_j w_j$$

$$y = f(y\_in)$$

- Back propagation
  - error calculating

$$\delta = (t - y) f'(y\_in)$$

  - weight update

$$\Delta w_j = \alpha \delta z_j$$

- bias update

$$\Delta w_0 = \alpha \delta$$

- Sum input error

$$\delta\_in_j = \sum_{m=1}^{m} \delta w_j$$

$$\delta_j = \delta\_in_j f'(z\_in_j)$$

- weight correction

$$\Delta V_{ij} = \alpha \delta_j x_i$$

- Bias correction

$$\Delta V_{0j} = \alpha \delta_j$$

- Update

$$w_j = w_j + \Delta w_j$$

$$v_{ij} = v_{ij} + \Delta v_{ij}$$

- Repeat for the prescribed # of epochs, then adjust alpha and/or select a new max # of epochs

Deployment:

- Present all patterns to the network
- Use the feed forward portion of the algorithm to find y for each pattern
- Apply bipolar step function to y

$$y = g(y)$$

- If y = target value for that pattern, report a 1 in the match matrix

- Report to console
  - V initial weights
  - W initial weights
  - V final "
  - W final "
  - learning rate
  - # epochs
  - match matrix

1 signifies a match for the pattern at that index, 0 is a miss

Test!

Results for all epoch/α combos are in Project 6.1 Results.xlsx

Analysis:

The network only reached 100% accuracy for 2 cases. Training for 25000 epochs with learning rate at .25 or .5 gave a full match. In both cases, the final weights were much larger than the initial weights.

From this experiment, we can say that an efficient and accurate solution is dependent on having a large enough learning rate and # of epochs.

Project 6.2:

Given:

Back propagation

1 hidden layer

bias on v and w

10 inputs

4 hidden

2 outputs

Bipolar sigmoid

Test set w/ 8 patterns

Heterogenous

$W = V = [-.5 .. 5]$ random

$\alpha = .05, .5$

5000 and 50000 epochs

Find:

- Responces and classification ability

Net Construction:

Back propagation, online learning

10 unit inputs, 2 unit outputs

Update weights, bias w/ general delta rule

V will be 10x4 matrix, w will be 4x2

Stopping condition will be 5000 and 50000 epochs

Bipolar sigmoid activation

Algorithm:

- Define $f(x)$ as bipolar sigmoid, $g(x)$ as bipolar
- Allocate test sets, weights, and variables
- Initialize w and V w/ random values
- select α, randomly select a training vector
- Train feed forward
    - Calculate z values

$$z\_in_j = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

$$z_j = f(z\_in_j)$$

- Calculate Y values

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

$$y = f(y\_in_k)$$

- Back propagation
  - error calculating

$$\delta_k = (t - y_k) f'(y\_in_k)$$

  - weight update

$$\Delta w_{jk} = \alpha \delta_k z_j$$

- bias updte

$$\Delta W_{0K} = \alpha \delta_K$$

- Sum inputerror

$$\delta\_in_j = \sum_{m=1}^{m} \delta_k W_{jK}$$

$$\delta_j = \delta\_in_j f'(z\_in_j)$$

- weight correction

$$\Delta V_{ij} = \alpha \delta_j X_i$$

- Bias correction

$$\Delta V_{0j} = \alpha \delta_j$$

- Update

$$w_{jK} = w_{jK} + \Delta w_{jK}$$

$$v_{ij} = v_{ij} + \Delta v_{ij}$$

- Repeat for the prescribed # of epochs, then adjust alpha and/or select a new max # of epochs

Deployment!

- Same as method for project 6.1

Test!

Results for all epoch/α combos are in Project 6.2 Results.xlsx

Analysis!

All cases resulted in a 100% match.

As demonstrated in problem 1, both W and V final values had some values that differed greatly from the initial weight values.

From this, we can say that it is possible to classify patterns with decimal values using the Back propagation Network. Also, we can diagnose

the effectiveness of training by seeing whether the weights for any neuron are changing or not.

Finally, we can also see that there are some cases where the required learning rate and # of epochs are small, as we got 100% accuracy from using a case with $\alpha = .05$ and 5000 epochs.

Project 6.3!

Given!

Backpropagation

5x3 inputs

1x3 outputs

no more than 15 hidden nodes

Find!

- Responses / classification ability
- Experiment
  - # hidden units
  - learning rate
  - W, V initial
  - Epochs
- Compare w/ Bam

Net construction!

Back propagation, online learning

15 unit inputs, 3 unit outputs

update weights/bias w/ general delta rule

Start w/ 1 hidden units, go until results stop changing

then test 15 hidden units

V will be 15 x # of Z units, W will be # of Z units x 3

Only stopping condition will be 25000 epochs to standardize results

learning rate will be .05, .25, .5

Will initialize V and W in bounds of random #s

First [-1, 1], [-.5, .5], [-.1, .1]

Bipolar sigmoid

Chose 25000 epochs because 5000 epochs proved to be ineffective in reaching 100% accuracy in project 6.1. Training for 50000 epochs yielded long program runtimes.

Algorithm:

- Same as Project 6.2, but a few edits
  - When allocating test patterns we will vectorize and reallocate in single S and t matrices
  - Will also build in ability to change initial weight bounds and # of hidden nodes

Deployment!

- Same as 6.1 and 6.2, but will also report # of hidden nodes and initial weight ranges

Test!

Results for all α/#z/initial weights combos are in Project 6.3 Results.xlsx

Each sheet has runs for a different initial weight range

Analysis:

We measured the network response to the following changing parameters

- Hidden nodes
  - Increasing the # of hidden nodes increased the # of successful matches, but also increased the runtime of the program. No benefit to accuracy was found past 4 nodes.

- learning rate
  - Increasing the learning rate did not really increase the successful match rate, but did affect which patterns successfully matched.
- Initial weight range
  - Changing the initial weight range had a similar effect to varying the learning rate.

Overall accuracy was not really affected, but the patterns that successfully matched did.

From this observed behavior, we can see that increasing the # of hidden units can result in higher accuracy but can also decrease the efficiency of the network.

In addition, changing the initial weight ranges and learning rate did not seem to improve accuracy. However, it does seem to change which neurons are trained first, although this could just be an effect of randomly presenting training vectors.

Bam Comparison!

Running the same data set of all letters from A to H on the BAM net yeilds 4/8 successful matches.

Running the same set on the Back-propagation net yields 8/8 succiess ful matches.

By using a Backpropagation net, we get a massive accuracy boost of +50%

Project 6.4!

Given!

Back propagation

1 hidden layer

Bias on V and W

Bipolar sigmoid

$y = \sin(2\pi x_1)\sin(2\pi x_2)$

$x_1 = [0, 1]$ range

$x_2 = [0, 1]$ range

Find!

- Responses / activation ability
- Experiment
  a) equally spaced training points, scrambled presentation
  10000, 1000 epochs
  ex 6.8
  b) isolate non zero targets
  c) randomly spaced training

Net construction a):

Back propagation, online learning

2 unit inputs, 1 unit outputs

Will vary between 5 and 10 hidden nodes

V will be 2 x # of z units W will be # of z units X 1

Stopping condition will be 1000, 10000 epochs

learning rate will be .25 to isolate testing of other parameters

Will initialize V and W in linear bounds first

For training

$X_1 = X_2 = [0, 1]$ steps of .2

For testing

" steps of .1

Bipolar sigmoid activation

Chose .25 learning rate because Project 6.1 showed that a value of .05 can yield innaccuracies regardless of the # of epochs.

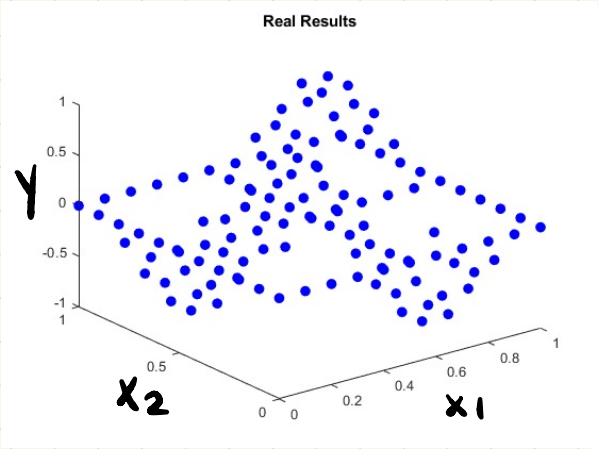
Algorithm!

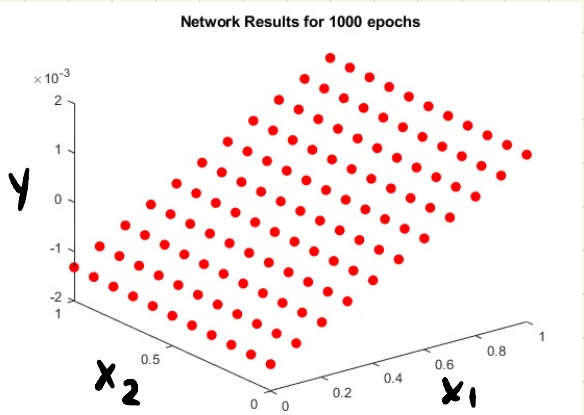
- same as project 6.2 but a few edits
- Will start off by defining X_1 and X_2 in [0, 1] w/steps of .2 and .1
- Will create a t set using the equation
- Testing all permutations of points
- Removing Bipolar step function
- Will display results in a table like EX 6.8 and in a 3D plot

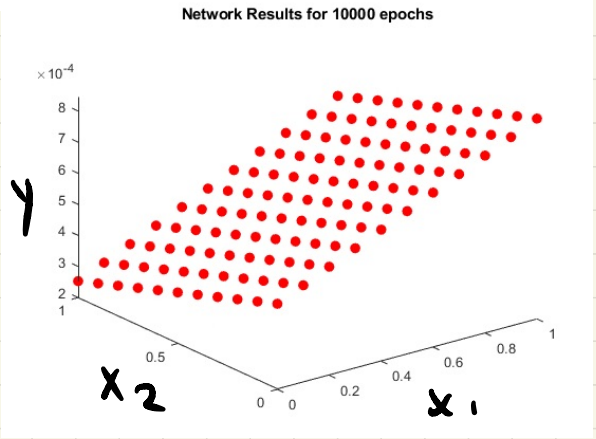
Test!

Results in Project 6.4 A folder

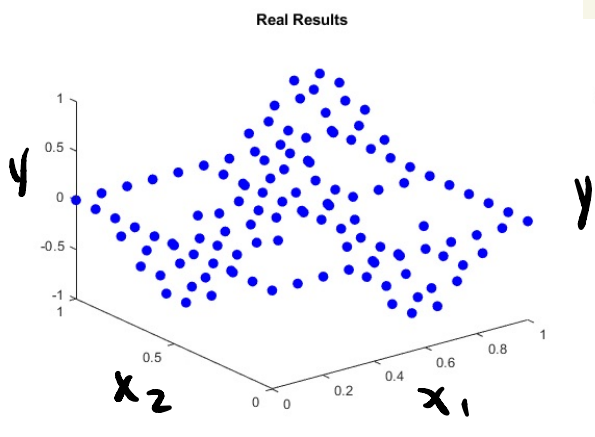
Plots for 5 hidden nodes!

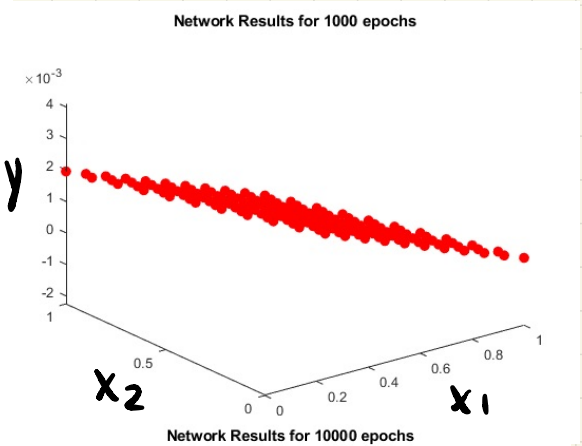


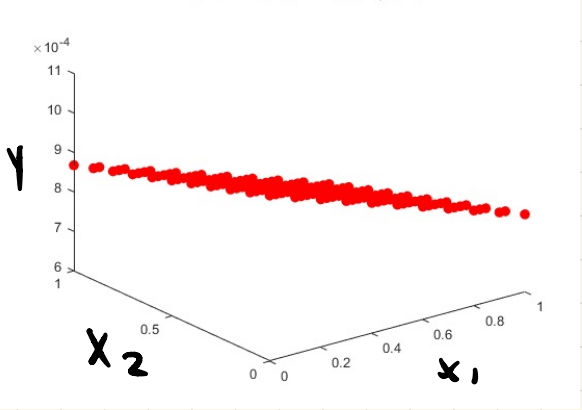




Plots for 10 hidden nodes:







Analysis:

First, all results seemed to converge to a plane rather than the actual function.

Increasing the epochs from 1000 to 10000 gave us results on a smaller order of magnitude (10^{-3} to 10^{-4}). Increasing the # of hidden nodes just seemed to result in a plane with a different orientation.

Net construction (b)!

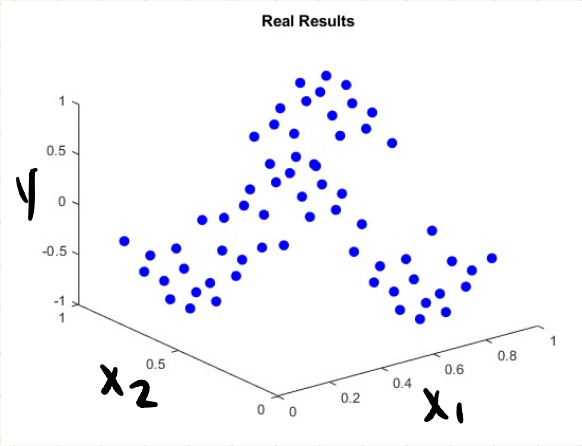
The implementation for portion B of the problem is the same as portion a) with one exception

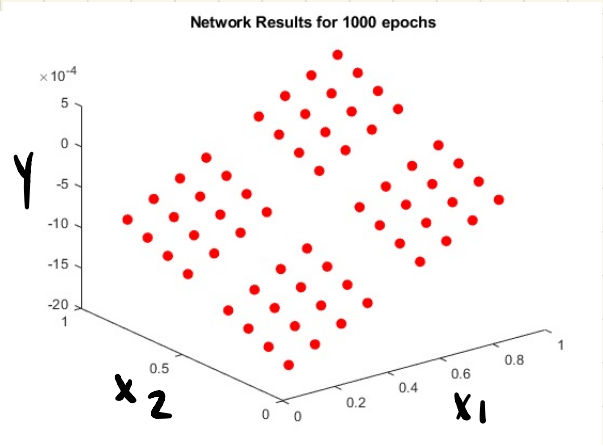
- All combinations with target values less than .05 will be removed from the training and test sets.

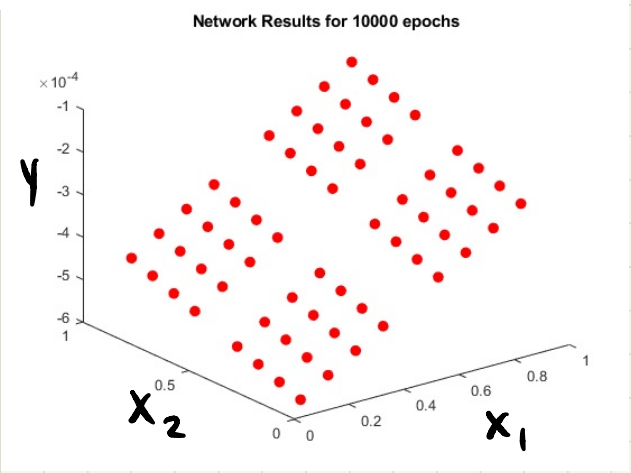
Test!

Results in Project 6.4 B folder

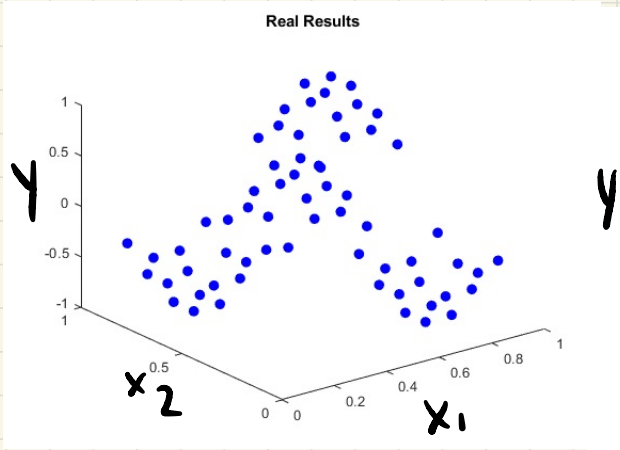
Plots for 5 hidden nodes!

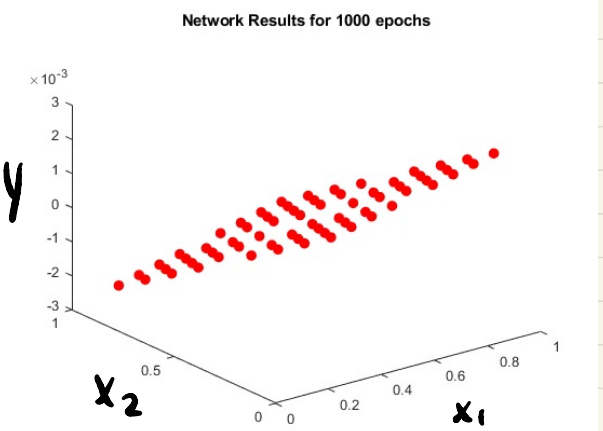


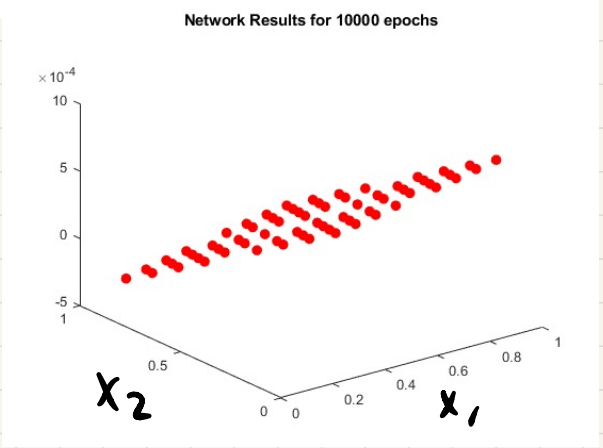




Plots for 10 hidden nodes!







Analysis:

Similar to part a), all results converged to a plane on order 10^{-3} for 1000 epochs and 10^{-4} for 10000 epochs.

Changing the number of epochs and hidden nodes have the same effects as part a).

Only real differences are that all points close to zero are removed from the plots/tables,

and that the plane orientation doesn't seem to change by increasing the # of epochs.

Net construction C)

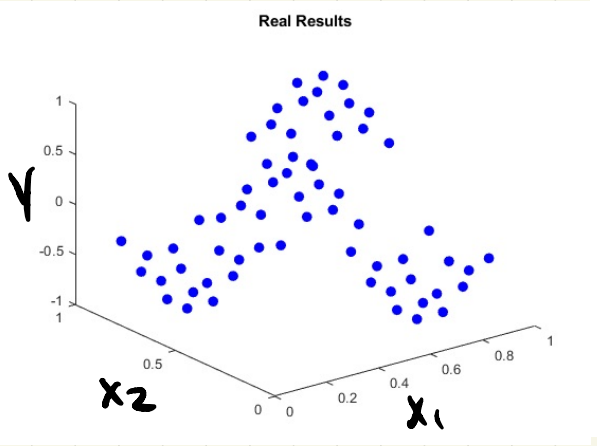
The implementation for portion C) of the problem is the same as portion B) with one exception

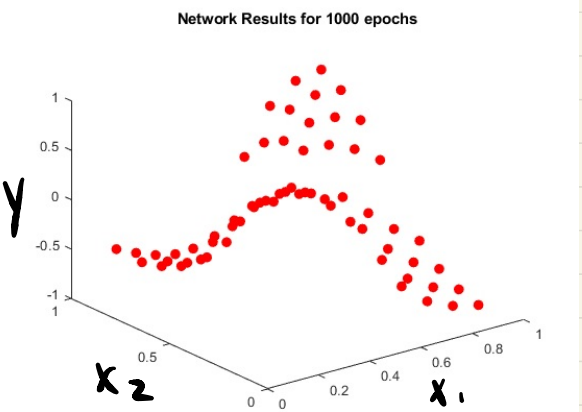
- Training data will be 20 randomized points rather than linearly spaced points.
- Test set will not change

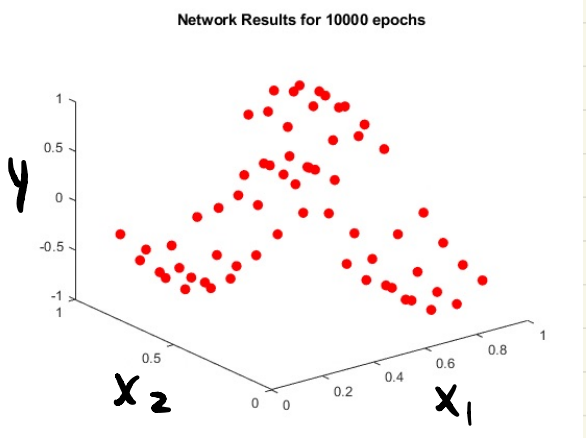
Test!

Results in Project 6.4C folder

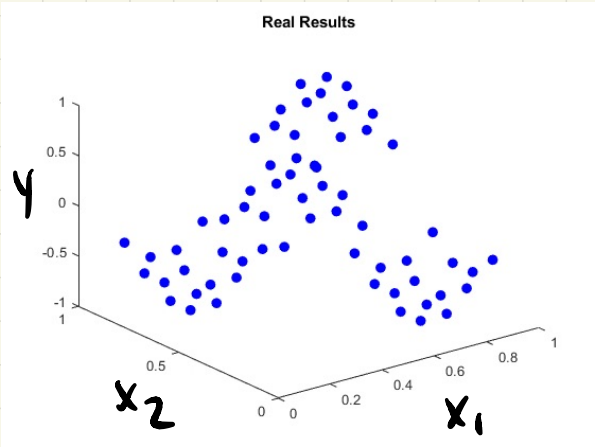
Plots for 5 hidden nodes!

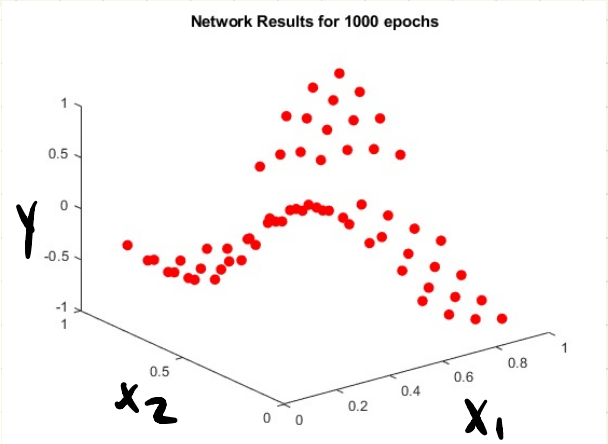


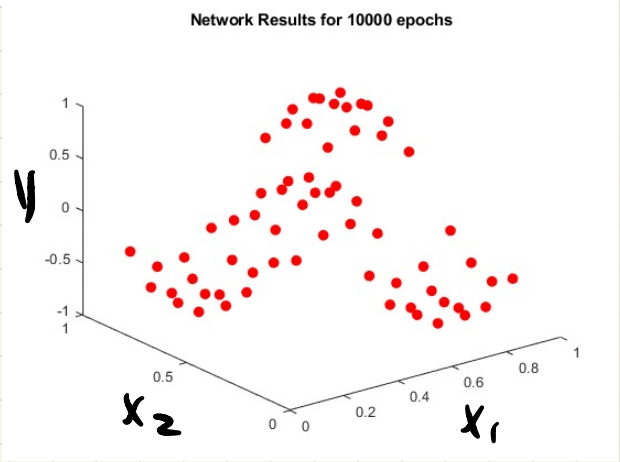




Plots for 10 hidden nodes!







Analysis:

Randomizing our training set yeilded a solution that actually converges to the function as shown in the plots.

Training for more epochs resulted in values closer to our real target values, while increasing the # of hidden nodes only seemed to increase program runtime.

This experiment showed how selecting the right training data for a network can severely impact performance. It seemed that by using linearly spaced training vectors, we were only fitting the network to a specific plane rather than a sine wave. It is likely that the linearly spaced training set introduced a bias to a plane in our network, and randomizing the inputs

removed that bias. By randomizing inputs and removing X_1, X_2 pairs with $t=0$, we can make our inputs orthogonal and free of bias towards and direction.