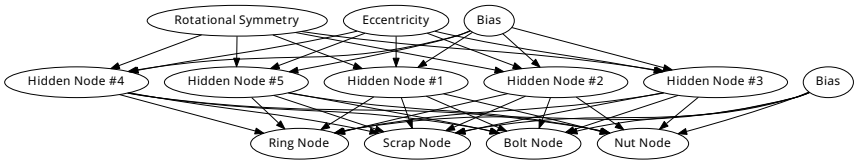
AI Project 2
Neural Nets

Dennis Honeyman & Cameron Burton

Spring Quarter

1 Network Design

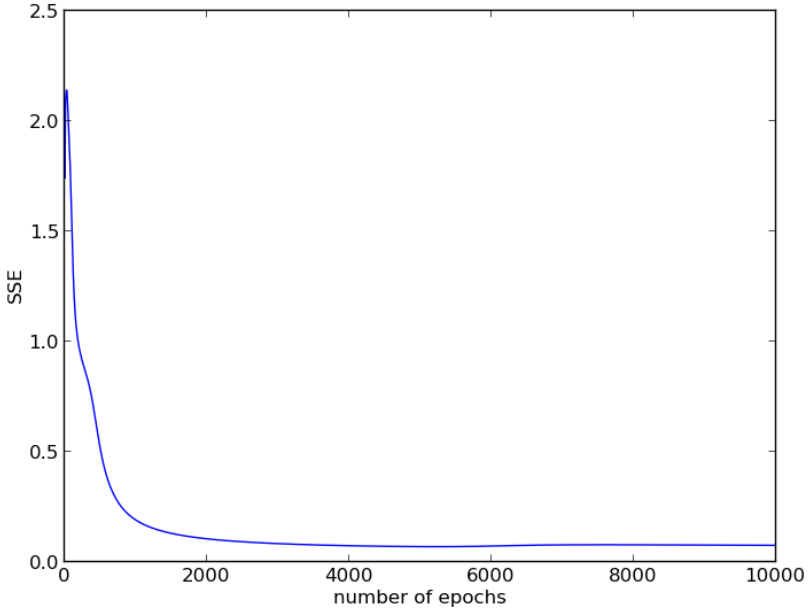


In any neural network there is a "sweet spot", where the sum of squared error is the lowest within a reasonable number of epochs. Depending on the complexity of the training data as well as its size this can vary greatly. We discovered that after about 500 epochs we were already fairly well trained for the test data that was provided. Somewhere between 1,000 and 10,000 epochs seemed optimal, mainly due to how long it took to run 10,000 epochs over the training data. The simplicity of the input data was to our favor. If the testing data was more complex, we could have possibly run into an over fitting problem.

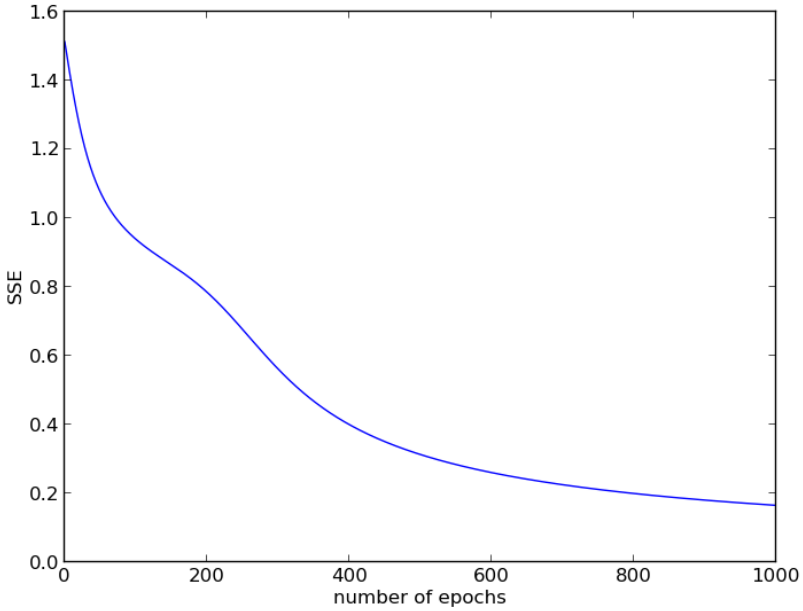
2 Data Sets and Results

2.1 1,000/10,000 Epochs

10,000


2.5
2.0
1.5
1.0
0.5
0.0
SSE
0
2000
4000
6000
8000
10000
number of epochs

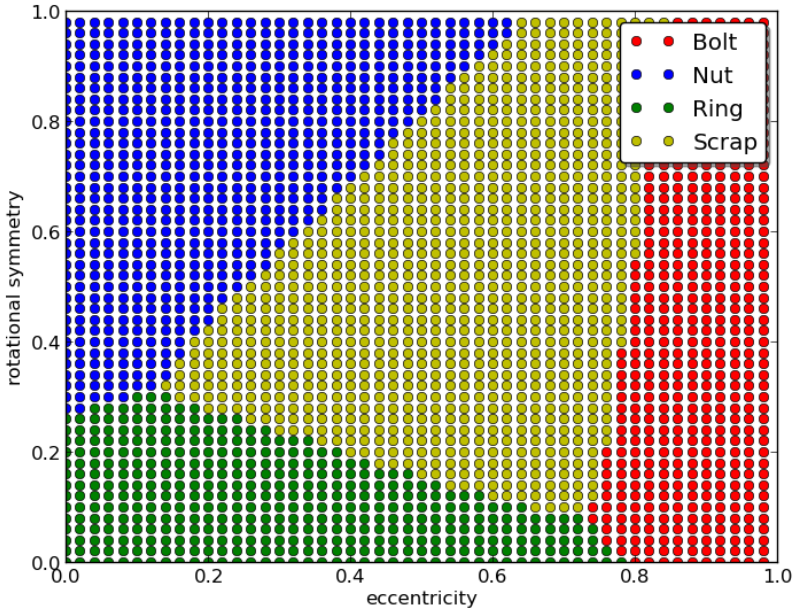
1,000


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
SSE
0
200
400
600
800
1000
number of epochs

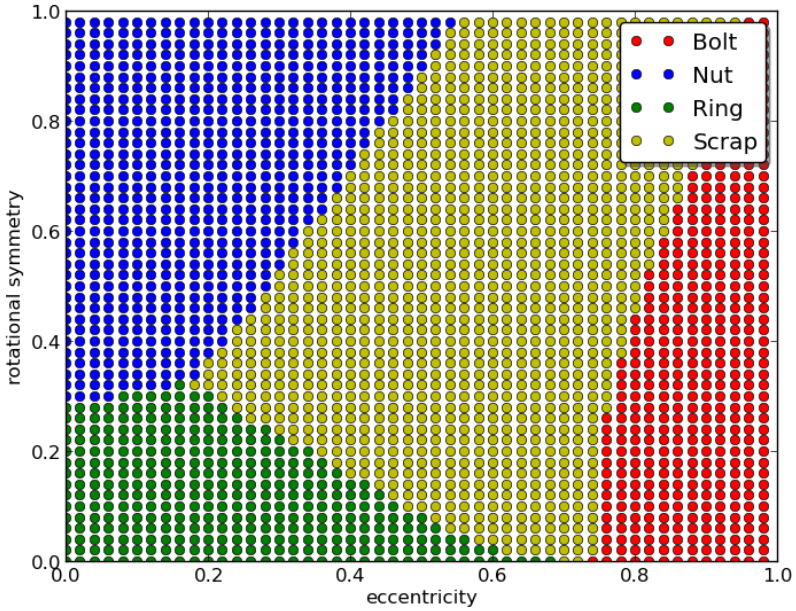
2.1.1 1,000/10,000 Epochs, `test_data.csv`

- Number of Errors: 0
- Percent Wrong: 0%
- Profit: $2.03

10,000 Epochs



1,000 Epochs



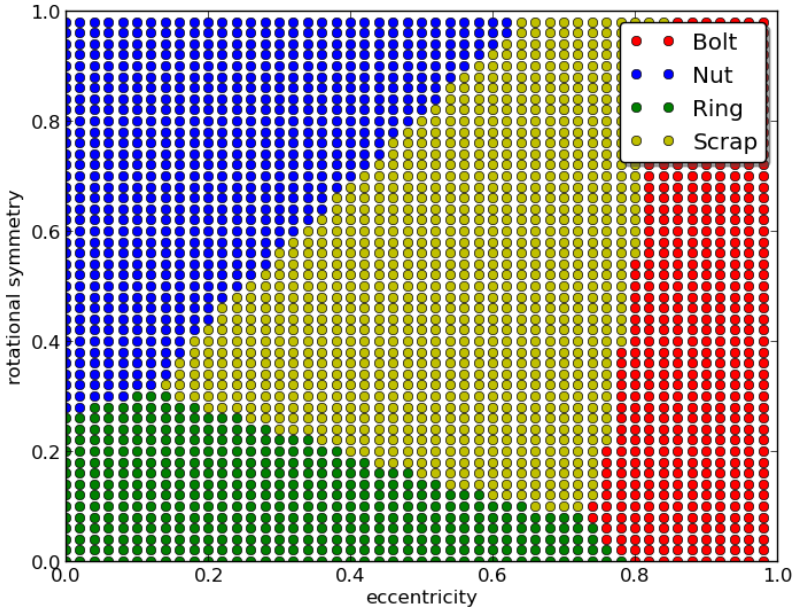
Output (top) vs. Expected (left)				
	1	2	3	4
1	5	0	0	0
2	0	6	0	0
3	0	0	5	0
4	0	0	0	4

Note: 10,000 Epoch data is exactly the same as 1,000 Epoch data.

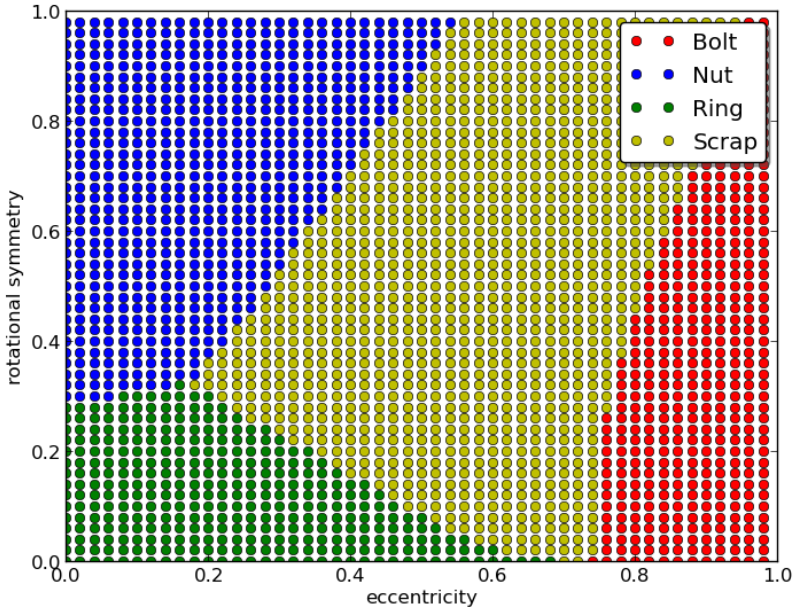
2.1.2 1,000/10,000 Epochs, `train_data.csv`

- Number of Errors: 3
- Percentage Wrong: 4.054%
- Profit: $6.64

10,000 Epochs



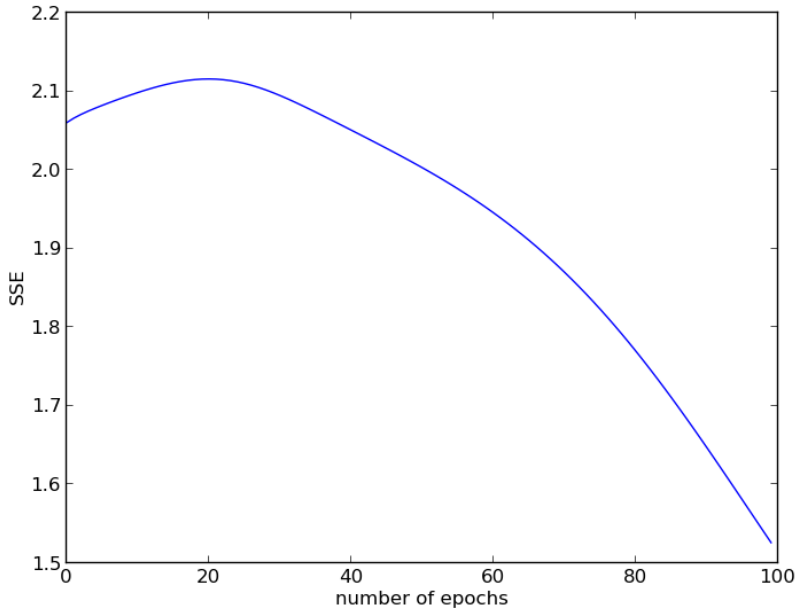
1,000 Epochs



Output (top) vs. Expected (left)				
	1	2	3	4
1	14	0	0	0
2	0	22	0	1
3	0	0	22	1
4	1	0	0	13

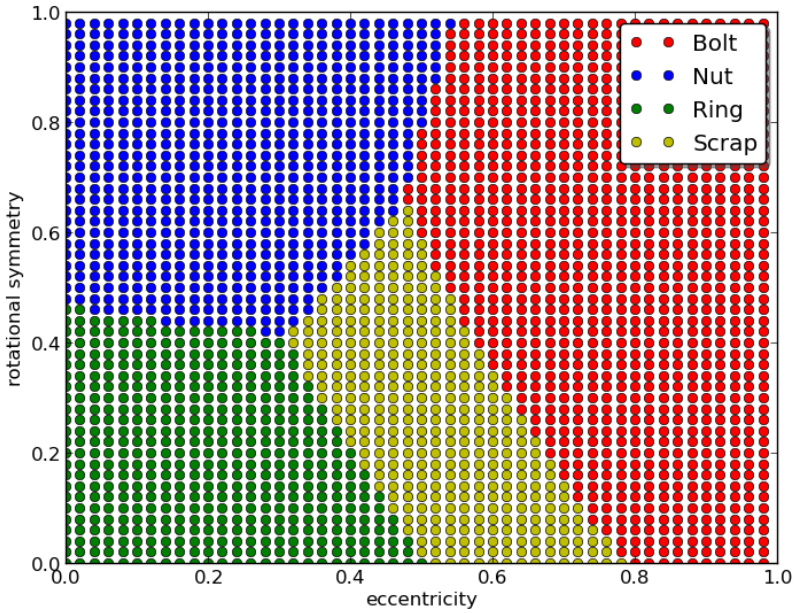
Note: 10,000 Epoch data is exactly the same as 1,000 Epoch data.

2.2 100 Epochs



2.2.1 100 Epochs, `test_data.csv`

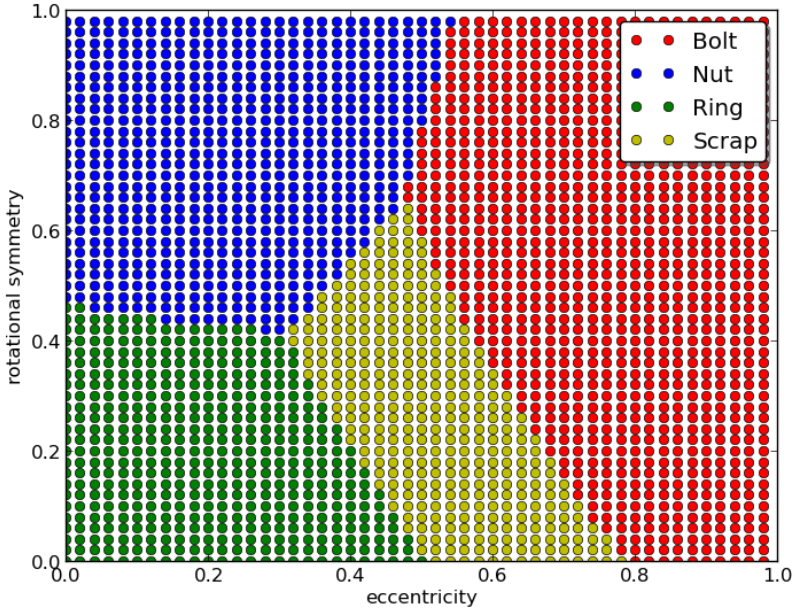
- Number of Errors: 6
- Percent Wrong: 30%
- Profit: $1.07



Output (top) vs. Expected (left)				
	1	2	3	4
1	5	0	0	1
2	0	2	0	1
3	0	4	5	0
4	0	0	0	2

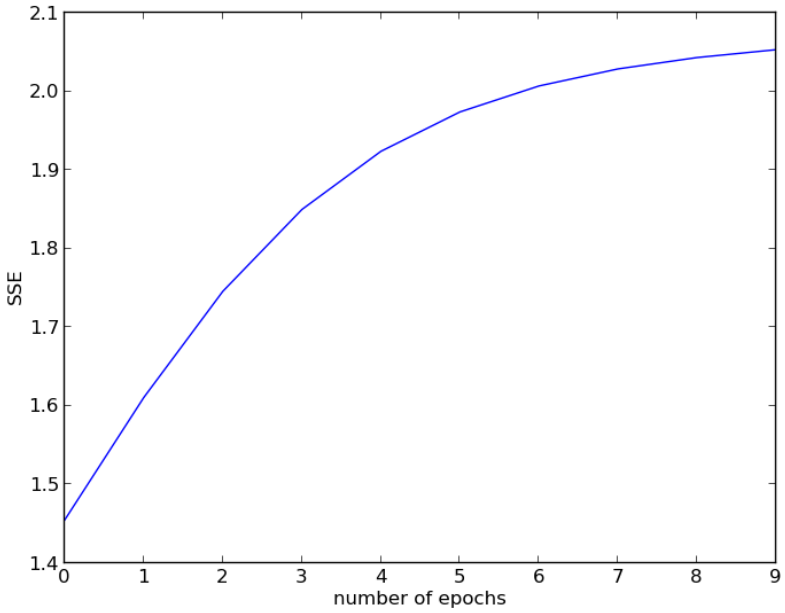
2.2.2 100 Epochs, `train_data.csv`

- Number of Errors: 19
- Percent Wrong: 25.676%
- Profit: $4.75



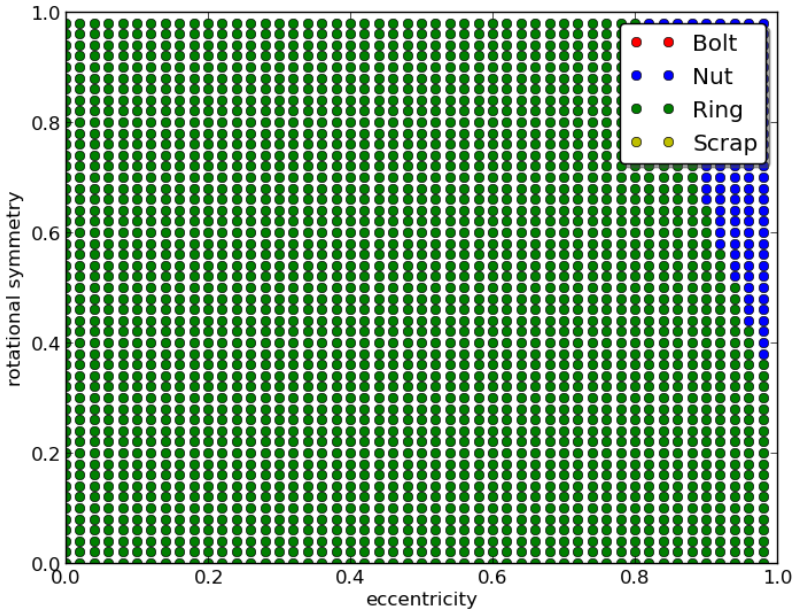
Output (top) vs. Expected (left)				
	1	2	3	4
1	15	0	0	7
2	0	14	0	0
3	0	8	22	4
4	0	0	0	4

2.3 10 Epochs



2.3.1 10 Epochs, `train_data.csv`

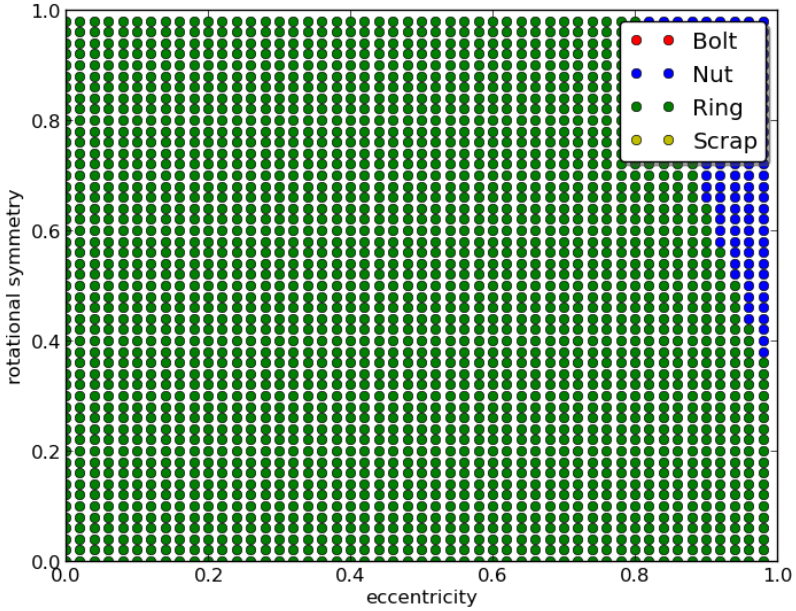
- Number of Errors: 52
- Percent Wrong: 70.27%
- Profit: $-2.54



Output (top) vs. Expected (left)				
	1	2	3	4
1	0	0	0	0
2	2	0	0	0
3	13	22	22	15
4	0	0	0	0

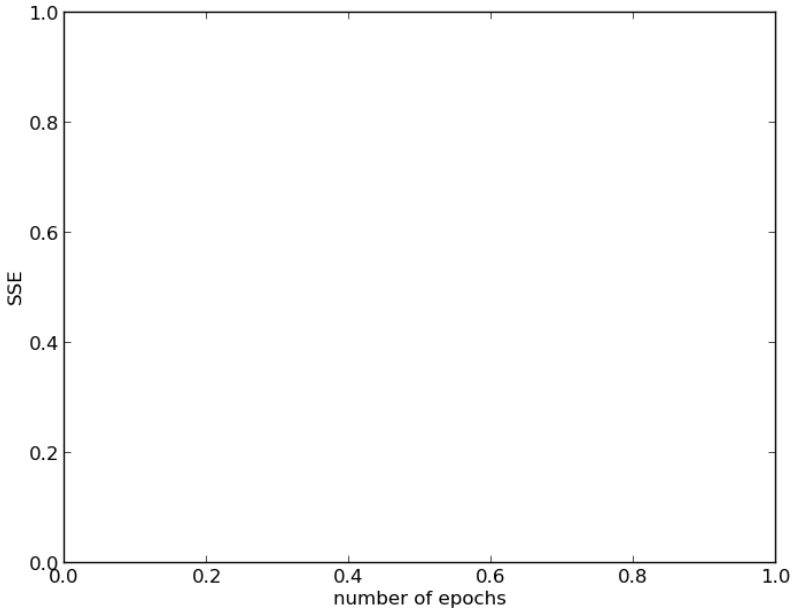
2.3.2 10 Epochs, `test_data.csv`

- Number of Errors: 15
- Percent Wrong: 75%
- Profit: $-0.80



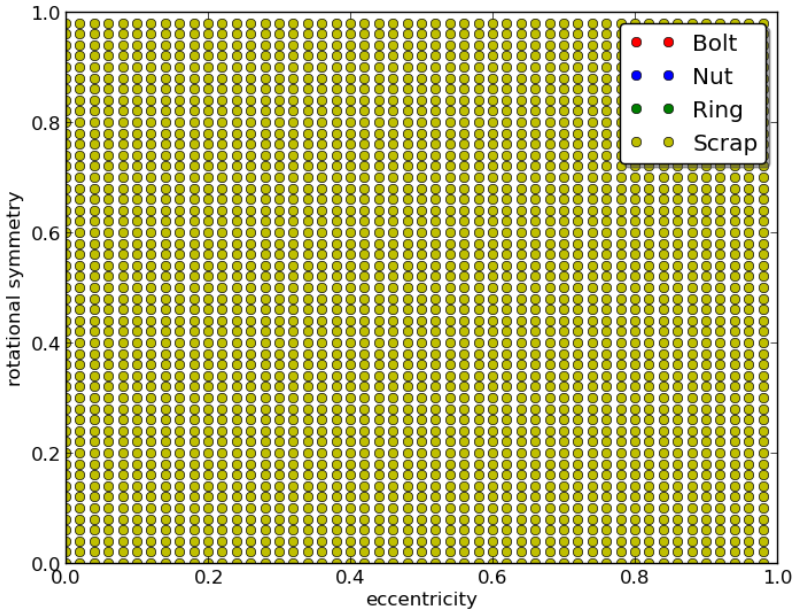
Output (top) vs. Expected (left)				
	1	2	3	4
1	0	0	0	0
2	1	0	0	0
3	4	6	5	4
4	0	0	0	0

2.4 0 Epochs



2.4.1 0 Epochs, `train_data.csv`

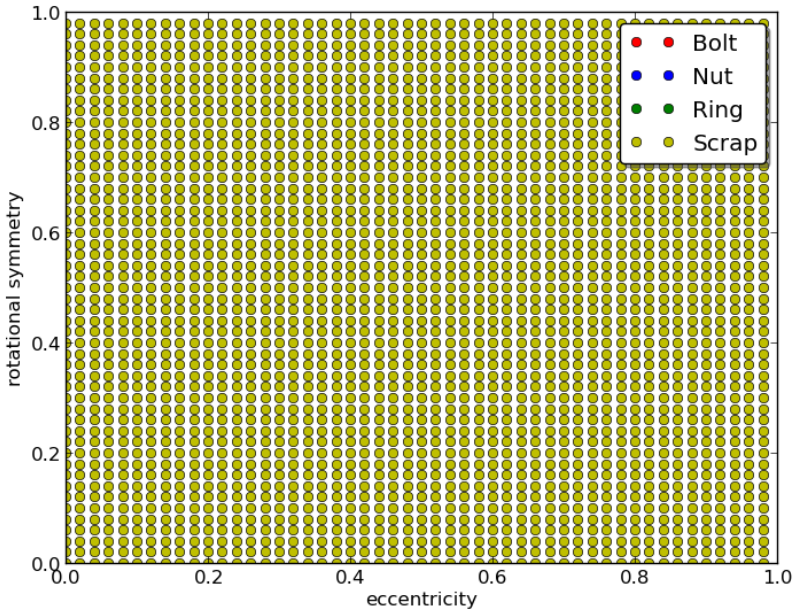
- Number of Errors: 59
- Percent Wrong: 79.73%
- Profit: $-2.22



Output (top) vs. Expected (left)				
	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	15	22	22	15

2.4.2 0 Epochs, `test_data.csv`

- Number of Errors: 16
- Percent Wrong: 80%
- Profit: $-0.60



Output (top) vs. Expected (left)				
	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	5	6	5	4

3 Discussion

Dennis believed that 100 epochs would be the best. He thought that having more than that would make the network over fit the training data. Cameron on the other hand believed that somewhere between 1,000 and 10,000 would be the best number.

After running all the test however there was no noticeable difference between the 1,000 epoch network and the 10,000 epoch network and both of these networks out-performed the other networks. For practicality we would pick the 1,000 epoch network as the best due to the significantly reduced training time. The most surprising result was the fact that 10,000 epoch network performed on the same level as the 1,000 epoch network. We assumed that the small number of errors that were seen when running the 1,000 network would be "fixed" after some more training iterations.

There is a possibility that a "sweet spot" exists 1,000 and 10,000 epochs

where no errors are generated on both of these sets of data. Finding this spot, however, might cost more time and energy than it would to use this "sub-optimal" network, profit loss included.

3.1 Implementation curiosities

In order to ensure that our network was functioning properly, we found an example of a neural network and the weights after one update. We then implemented that network and ran it to ensure we were getting the same results.