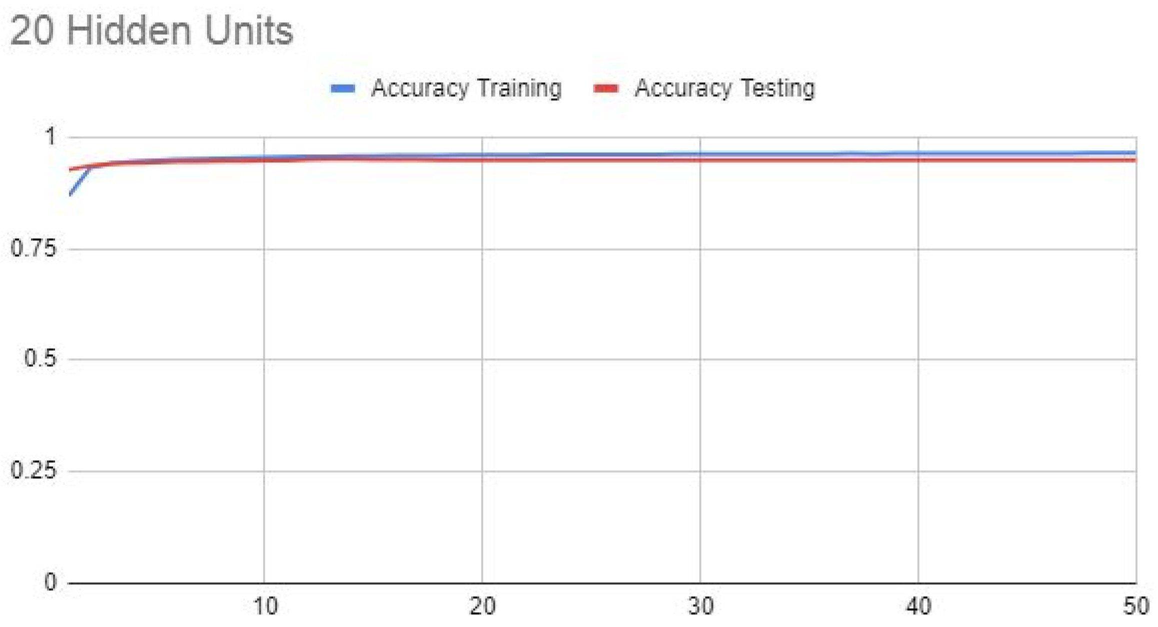
Peter Bui ID: 973296176

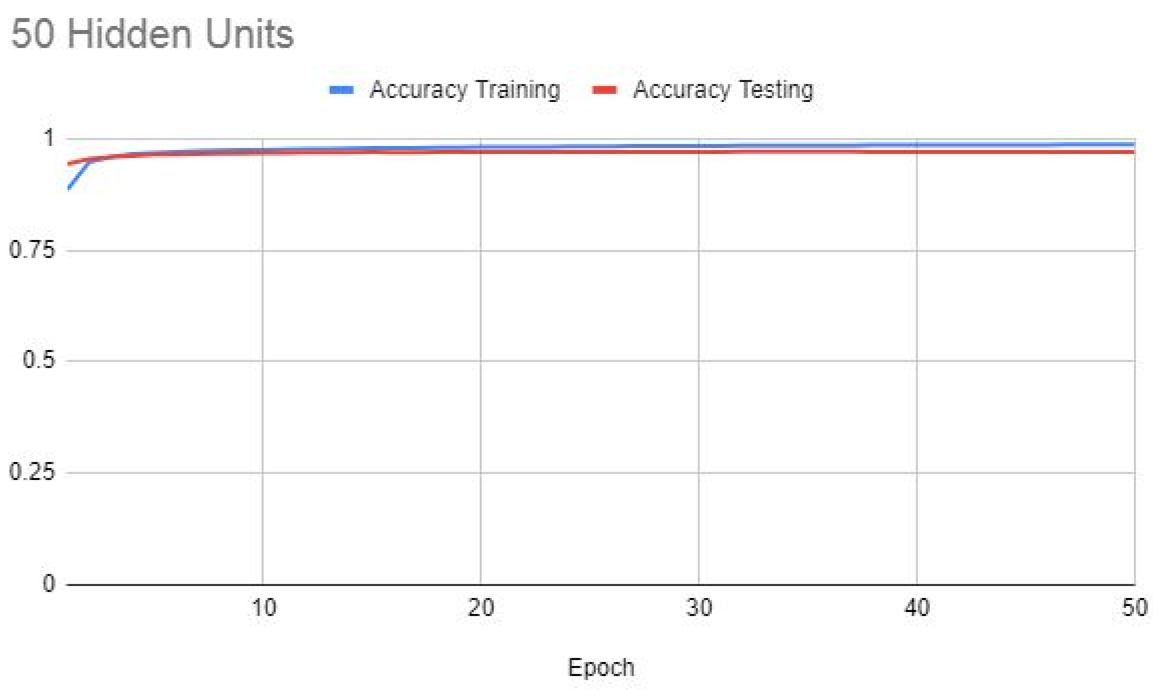
Experiment 1

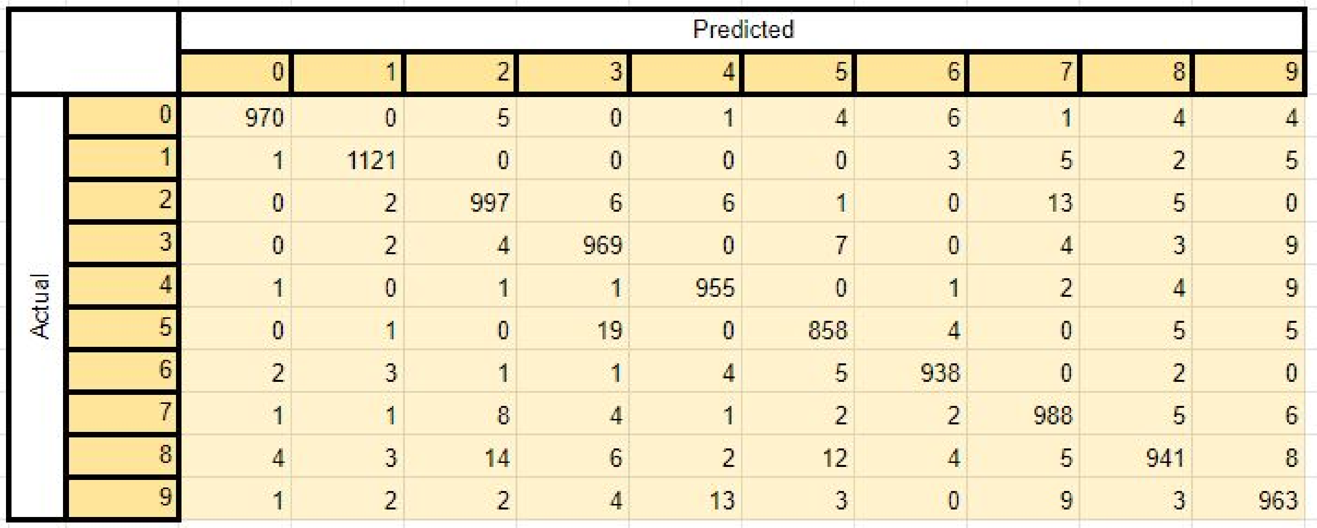
Description:

This experiment takes two data sets, mnist_train.csv, and 'mnist_test.csv. They contain pixel values corresponding to images of handwritten digits. The purpose of the experiment is to train a neural network consisting of a group of 10 perceptrons and three different amounts of hidden units (20, 50, 100) to correctly identify each example's 784 inputs. This time, the learning rate is set to .1. Starting with randomized weights, they are adjusted over the course of 50 epochs as the neural network calculates them with a given input. The weights change regardless if it matches the target output, but using .1 and .9 allows the weights to change in favor of the target label. Predictions are stored by taking the largest calculated value out of the 10 outputs of the neural network, and their accuracy is put against the total amount of predictions. A confusion matrix is also utilized to see the perceptron's choice versus the correct choice. This shows which digits are commonly misinterpreted.



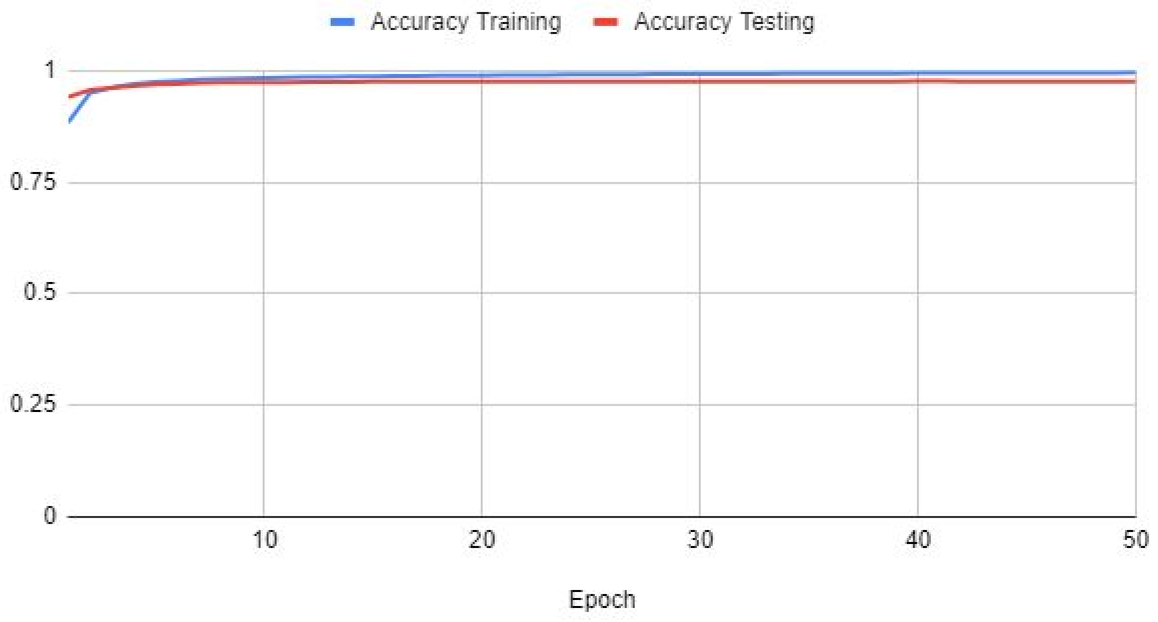
		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	958	0	5	0	1	9	11	0	9	4
	1	0	1110	0	1	0	1	2	7	3	1
	2	0	6	967	18	3	1	1	17	4	1
	3	3	1	9	948	1	20	1	6	3	10
	4	2	1	4	2	944	3	4	4	6	40
	5	4	1	5	13	2	815	12	0	7	6
	6	6	4	16	0	9	12	916	0	8	2
	7	2	2	7	10	1	4	0	973	3	6
	8	5	10	19	12	3	19	11	5	926	10
	9	0	0	0	6	18	8	0	16	5	929





		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	970	0	5	0	1	4	6	1	4	4
	1	1	1121	0	0	0	0	3	5	2	5
	2	0	2	997	6	6	1	0	13	5	0
	3	0	2	4	969	0	7	0	4	3	9
	4	1	0	1	1	955	0	1	2	4	9
	5	0	1	0	19	0	858	4	0	5	5
	6	2	3	1	1	4	5	938	0	2	0
	7	1	1	8	4	1	2	2	988	5	6
	8	4	3	14	6	2	12	4	5	941	8
	9	1	2	2	4	13	3	0	9	3	963

100 Hidden Units



		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	971	0	4	1	1	3	7	0	3	4
	1	0	1116	2	0	0	0	3	1	1	3
	2	0	2	1003	4	3	0	1	11	2	0
	3	0	4	5	994	0	9	1	3	4	9
	4	0	1	1	0	952	1	1	1	4	5
	5	0	3	0	2	0	863	7	0	3	3
	6	3	1	4	0	4	8	933	0	2	1
	7	2	2	10	3	2	2	1	1005	4	6
	8	3	6	3	3	1	3	4	1	949	8
	9	1	0	0	2	19	3	0	6	2	970

1. How does the number of hidden units affect the final accuracy on the test data?

   The number of hidden units affects the final accuracy on the test data by increasing the rate at which the neural network learns. The more there are, the faster it can get to more accurate predictions

2. How does it affect the number of epochs needed for training to converge?

   The number of hidden units does not seem to affect the number of epochs needed for the training set to converge with the testing set

3. Is there evidence that any of your networks has overfit to the training data? If so, what is that evidence?

   There is a little bit of small oscillations in the training data which can indicate overfitting. This can either be attributed to a higher number of hidden units or the neural network relying on past weights

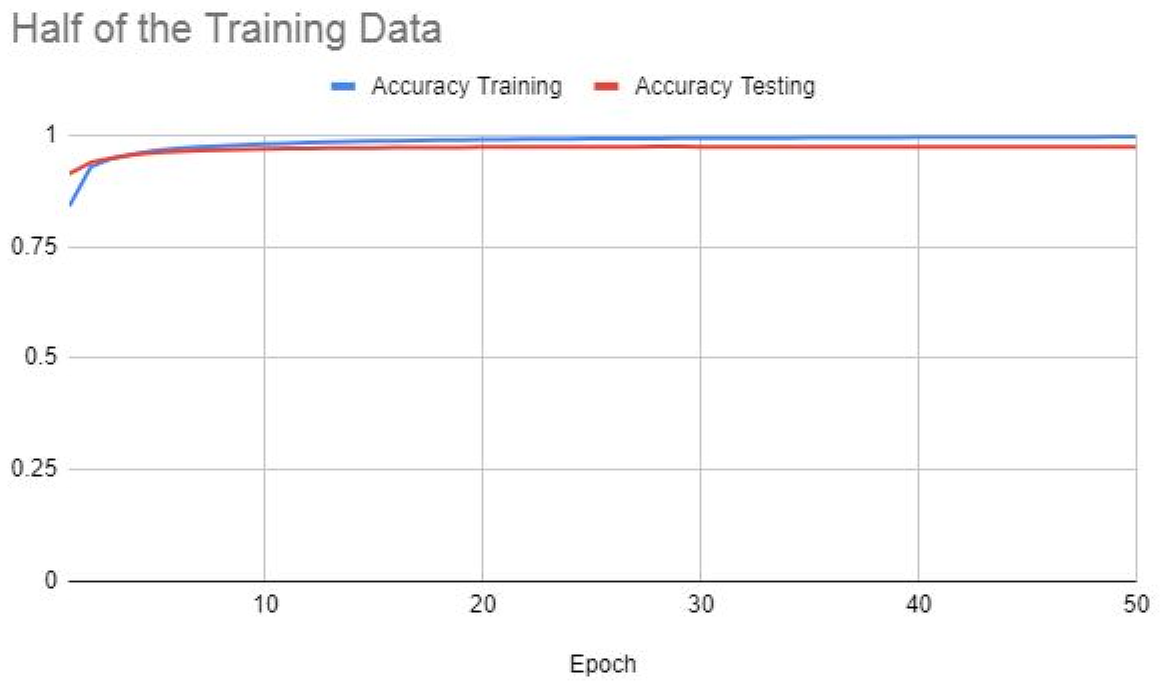
4. How do your results compare to the results obtained by your perceptron in Assignment 1?

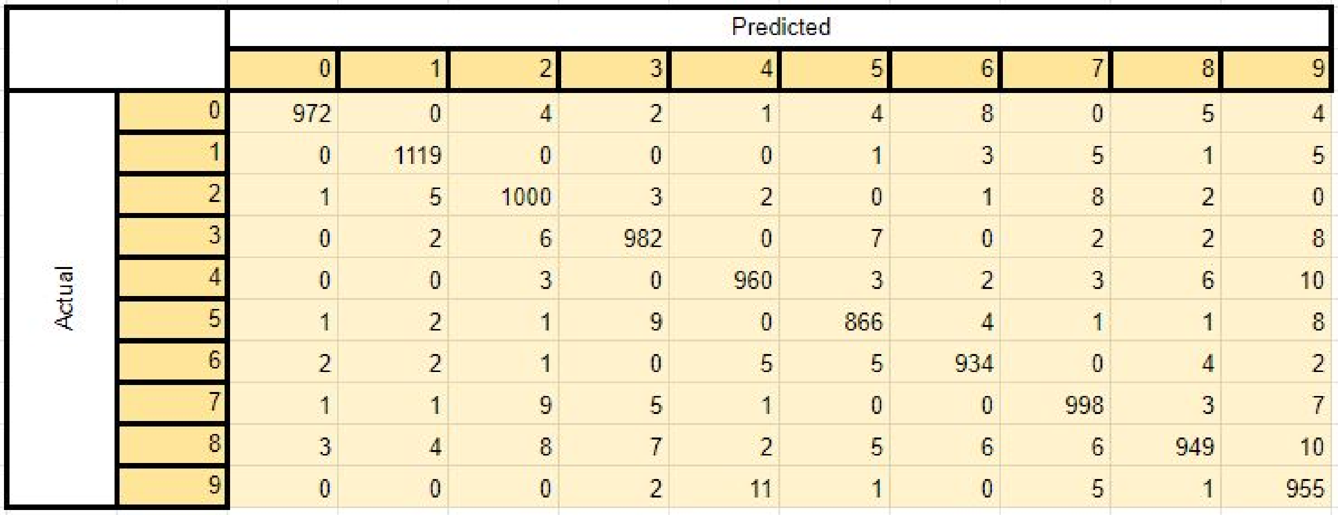
   Overall, there are less oscillations in these results than in Assignment 1, which could be attributed to the use of a sigmoid function instead of a step function

Experiment 2

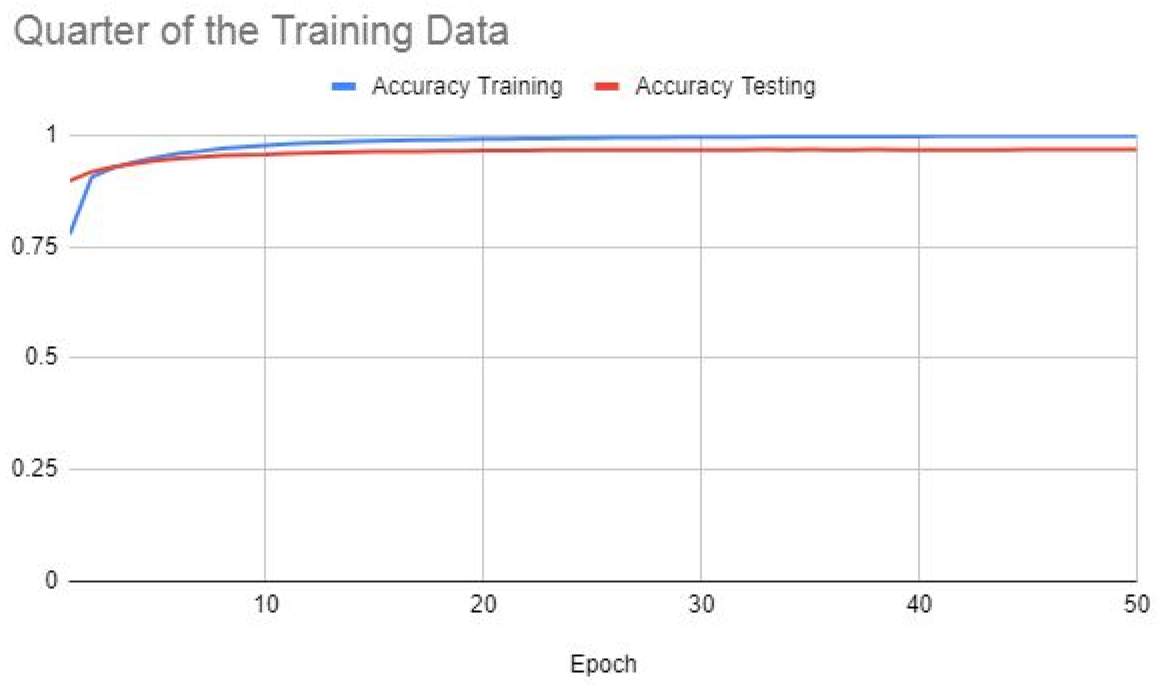
Description:

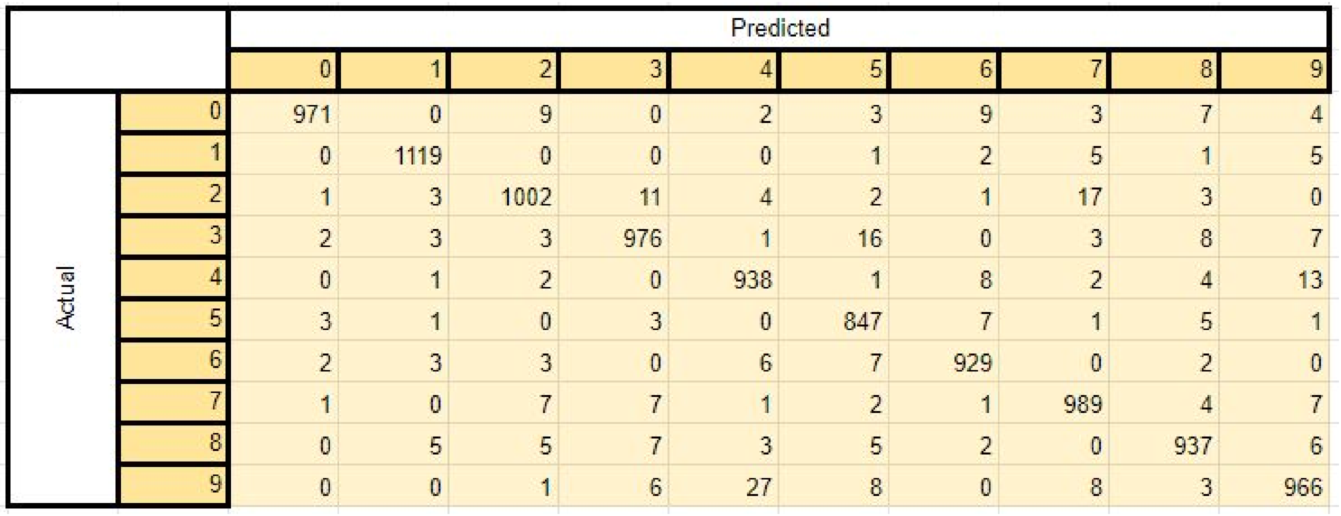
This experiment sets the number of hidden units to 100 and only varies the amount of training data supplied. Half of the original training data (30,000 examples) and a quarter (15,000 examples) are used. To ensure that the amount of classes are balanced, the original set is shuffled and then sliced.





		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	972	0	4	2	1	4	8	0	5	4
	1	0	1119	0	0	0	1	3	5	1	5
	2	1	5	1000	3	2	0	1	8	2	0
	3	0	2	6	982	0	7	0	2	2	8
	4	0	0	3	0	960	3	2	3	6	10
	5	1	2	1	9	0	866	4	1	1	8
	6	2	2	1	0	5	5	934	0	4	2
	7	1	1	9	5	1	0	0	998	3	7
	8	3	4	8	7	2	5	6	6	949	10
	9	0	0	0	2	11	1	0	5	1	955





		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	971	0	9	0	2	3	9	3	7	4
	1	0	1119	0	0	0	1	2	5	1	5
	2	1	3	1002	11	4	2	1	17	3	0
	3	2	3	3	976	1	16	0	3	8	7
	4	0	1	2	0	938	1	8	2	4	13
	5	3	1	0	3	0	847	7	1	5	1
	6	2	3	3	0	6	7	929	0	2	0
	7	1	0	7	7	1	2	1	989	4	7
	8	0	5	5	7	3	5	2	0	937	6
	9	0	0	1	6	27	8	0	8	3	966

1. How does the size of the training data affect the final accuracy on the test data?
   The size of the training data causes the final accuracy of the test data to start lower as the number of examples is lowered. This makes sense since there are less examples to draw conclusions from so accuracy for the first few epochs would be lower

2. How does it affect the number of epochs needed for training to converge?
   The amount of training examples does not appear to affect the number of epochs needed for the training set to converge with the testing set either

3. Again, is there evidence that any of your networks has overfit to the training data? If so, what is that evidence?
   There is still some overfitting which can be seen by small bumps in the accuracy

Conclusions:

Using a hidden layer on top of perceptrons makes the results much smoother, this may also be attributed to the sigmoid function. Confusion matrices are an improvement from the last assignment using numpy's set_printoptions. Having and starting with the small example with preset results given in class was a big help in getting the algorithm down, though there were a couple places where results had to be rounded to keep the numbers the same, otherwise the class example was very helpful.