Louis Vu
Homework4
Problem 3, Extra Credit

I created a text file that held dataset of handwritten digits from values zero to nine approximately 1200 examples big. I also created a new test set approximately 400 examples big that output values zero to nine. My accuracy came out to be approximately 90 percent when parameters are as given: 20 .01 100 trainEC.txt testEC.txt.

I wrote two other activation functions, tanh and sigmoid, and noticed that the sigmoid function along with the derivative of the sigmoid function will give similar accuracy as ReLU, but tanh function along with the derivative of the tanh function gives a slightly lower accuracy.

I wrote a back propagation algorithm that uses average gradient for all training examples when updating weights and noticed the accuracy went way down. It is possible that I may have implemented the code incorrectly.

```
// tanh activation function g(x)
	outputValue = Math.tanh(sum);

// sigmoid activation function g(x)
	outputValue = 1/(1 + Math.exp((-1)*sum));

/*gPrime for delta of hidden nodes*/
// tanh gPrime
	double gPrime = 1 - Math.pow(Math.tanh(hiddenNodes.get(r).getOutput()), 2);

// sigmoid gPrime;
double gPrime = hiddenNodes.get(r).getOutput()*(1-hiddenNodes.get(r).getOutput());

/*gPrime for delta of output nodes*/
// tanh gPrime
	double gPrime = 1 - Math.pow(Math.tanh(outputNodes.get(n).getOutput()), 2);

//sigmoid gPrime
double gPrime = outputNodes.get(n).getOutput()*(1-outputNodes.get(n).getOutput());

/* average gradient */
// average weights btwn input and hidden
	for(int s = 0; s < hiddenNodes.size()-1; s++){
		for(int t = 0; t < hiddenNodes.get(s).parents.size()-1; t++){
					hiddenNodes.get(s).parents.get(t).weight =
hiddenNodes.get(s).parents.get(t).weight/trainingSet.size();
				}
			}
// average weights btwn hidden and output
	for(int u = 0; u < outputNodes.size(); u++){
		for(int v = 0; v < outputNodes.get(u).parents.size()-1; v++){
					outputNodes.get(u).parents.get(v).weight =
outputNodes.get(u).parents.get(v).weight/trainingSet.size();
				}
```

```
                    }
//
      for(int s = 0; s < hiddenNodes.size()-1; s++){
            for(int t = 0; t < hiddenNodes.get(s).parents.size()-1; t++){
                              hiddenNodes.get(s).parents.get(t).weight +=
hiddenNodes.get(s).parents.get(t).node.getOutput()*learningRate*hidDelt[s];
                        }
                    }
// Update weight btwn hidden and output
      for(int u = 0; u < outputNodes.size(); u++){
            for(int v = 0; v < outputNodes.get(u).parents.size()-1; v++){
                              outputNodes.get(u).parents.get(v).weight +=
outputNodes.get(u).parents.get(v).node.getOutput()*learningRate*outDelt[u];
                        }
                    }
```