CS30 - Assignment 6

Due Friday the 11th, at 6pm

For this assignment, we're going to be doing some experiments with a neural network package implemented in python. Before starting this, make sure that you've read sections 1, 2 and 4 of the *Neural Nets in Python* handout posted on the course web page.

For this assignment, you may (and in fact, I'd encourage you to) work with a partner. If you work with a partner, you only need to submit one file, just make sure both of your names are at the top of it. **If you choose to do work with a partner, you must both be there whenever you're working on the assignment and you should work on each of the problems collaboratively.** In particular, what is not appropriate is to divide the problems up and then combine your answers into a single submission (or, of course, just have one person do all the work :).

1 High-level overview

This assignment consists of three parts. Think of each part as an experiment. Compare the results of several trials as you vary the number of hidden nodes, the learning rate, and the momentum factor. For each part, you should include a writeup of your experiments as a comment (in a Python comment). Comment on at least three of the following things in each writeup:

– some basic results, i.e. training error rate and testing error rate

- how well the networks perform with respect to the speed of convergence
- how reliable (or repeatable) are the results
- do the weights of the nodes tell you anything about the problem/input?
- other

This assignment is as much about reflecting on the experiments as doing them, so take time to write up what you have done (**this will be a major part of your grade on this assignment**).

2 Problems

Put your code work along with your comments in a file named with your first name and last name followed by `assign6.py` (or just both last names if there are two of you).

To use the neural network module, you will need to:

- Download and save the `cs30neural.py` file in the *same location as where your assign6.py file is.*
- At the top of your `assign6.py` file, include:

  `from cs30neural import *`

Work through each of the experiments below.

1. Train a network to compute the square function on numbers between 0.0 and 1.0.

   (a) Train and test networks using the training data on the left and the testing data below.

   (b) Repeat the experiment with the training data on the right.

Training Data

input	output
0.0	0.00
0.1	0.01
0.2	0.04
0.3	0.09
0.4	0.16
0.5	0.25

input	output
0.0	0.00
0.2	0.04
0.4	0.16
0.6	0.36
0.8	0.64
1.0	1.00

Testing Data

input	output
0.3	0.09
0.5	0.25
0.7	0.49
0.8	0.64
0.9	0.81

Commentary: In a sense, these experiments are unfair. When we use the training data on the left, we are expecting the network to *extrapolate* from the lower half interval to the upper half. With the training data on the right, we hope that the network will *interpolate* between the known values.

The experiments (and others on this assignment) are also unrealistic, in the sense that we already know the underlying function. There is no need to train a neural network to compute it. The value of the experiments is to give us a sense of how confident we can be in more realistic cases when we want the network to compute an unknown function.

What to include in your `.py` file for this problem:

- Your declarations in python of the different training data sets and testing data set.
- A few example calls showing some of the experiments that you ran.
- A 1-2 paragraph writeup discussing aspects of your experiments. Make sure to provide concrete results on the two different training data sets.

2. For this problem, we examine a similar experiment that uses the "greater than" relation.

   (a) Train a network with only the positive information in the training data on the left. Test the network on the testing data below.

   (b) Repeat the experiment using *both* sets of training data.

   (c) Train the network on all pairs of inputs taken from 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0. There will be 36 training "facts." Interpret "greater than" in the strict sense: The output is 0 when the inputs are equal to one another. Instead of typing all the training data, write a few lines of code to "generate" it (this will be graded as part of your submission).

Training Data

inputs		output
0.2	0.0	1
0.4	0.2	1
0.6	0.4	1
0.8	0.6	1
1.0	0.8	1

inputs		output
0.0	0.2	0
0.2	0.4	0
0.4	0.6	0
0.6	0.8	0
0.8	1.0	0

Testing Data

inputs		output
0.0	0.6	0
0.3	0.4	0
0.3	0.2	1
0.8	0.8	0
0.5	0.5	0

What to include in your `.py` file for this problem:

- Your declarations in python of the two basic training data sets and testing data set.
- Code that generates the training data consisting of all "facts".
- A few example calls showing some of the experiments that you ran.
- A 1-2 paragraph writeup discussing aspects of your experiments. Make sure to provide concrete results using the three different training data sets.

3. Finally, define a small classification problem with at least 10 training cases (or examples) and 5 testing cases. For inspiration, think of something in the realm of the election example from the lab. Train and test a neural network on this data and analyze its accuracy. Explain your intuition on why it did or did not work well.

   What to include in your `.py` file for this problem:

   - Your declarations in python of your training and testing data sets.
   - A few example calls showing some of the experiments that you ran.
   - A 1-2 paragraph writeup:
     - A sentence or two describing your data set and what it represents.
     - Results and and analysis of your experiment.

3 When you're done

Make sure that your submission is properly commented. In particular:

- You should have comments at the very beginning of the file stating your name, course, assignment number and the date.
- You should have comments delimiting the three problems.
- You should have concise, but thorough discussions of your experiments and what you found.

Submit your .py file online using the courses submission mechanism.

Grading

	points
`Q1`	
experiment code	2
analysis/writeup	3
`Q2`	
experiment code	2
analysis/writeup	3
function to generate examples	1
`Q3`	
data set	2
experiment code	2
analysis/writeup	3
total	18