CS 4633/6633 Third programming Assignment Due Nov. 20

This assignment is designed to help you understand how the backpropagation training algorithm for multi-layer feedforward neural networks works. It has two required parts and one part for extra credit.

Part 1: Gradient descent

The first part of the assignment is a simple exercise to help you understand the idea of gradient descent. You will use gradient descent to find the minimum of a simple function. The function must be continuous and differentiable, so that you can compute a derivative.

First, graph the function: $f(x) = (x - 1)^2$. What is its derivative?

The following simple C program uses gradient descent to find the minimum of this function, that is, the value of x for which y = f(x) has its minimum value.

In the program, the variable alpha is the learning rate. Run this program, and experiment with changing alpha to see what happens. How does changing alpha affect convergence? What would be a reasonable test for convergence? Then modify the program so that it finds the minimum of another function (your choice). Write a one-page description of your results, including simple graphs of the functions.

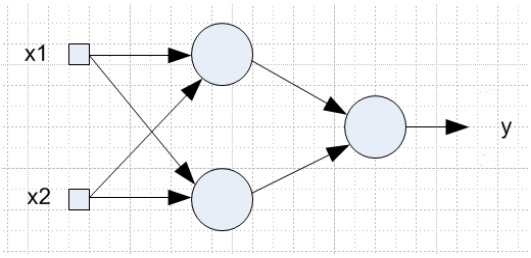
```
/* Let's find the minimum of f(x) = (x – 1)^2 using gradient descent */
#include (stdio.h)

float alpha=0.25, x=0.1, y ;
int   iter=0 ;

float f(float x) {
  return (x-1)*(x-1) ;
}
float derivative(float x) {
  return 2 * (x-1) ;
}

void main(void) {
  do {
      iter++ ;
      y = f(x) ;
      printf ( "%5d x=%6.3f y=%6.3f\n", iter, x, y ) ;
      x -= alpha * derivative(x) ;
  } while (iter < 20) ;
}
```

Part II: Backpropagation learning of the XOR function

The next part of the assignment is to write a program that simulates a multi-layer feedforward neural network and uses the backpropagation algorithm to train the network to learn the XOR function. Recall that the XOR function is the simplest example of a nonlinear function that a perceptron *cannot* learn. As the following table shows, there are only four possible inputs to the XOR function. Therefore, you will train the network on just these four examples.

Input		*Desired output*
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

For this problem, your neural network will have two input nodes and a single output node. It is sufficient to have a single hidden layer that contains one or two nodes, such as the following network.



Note that each node also has a bias input that is not shown in the diagram. The activation function for each node should be the sigmoid function. The logical interpretation of the network output is determined by the level of the output node's activation function, as follows:

1 if sigmoid(v) >= 0.8
0 if sigmoid(v) <= 0.2

Each epoch of training consists of training the network on the four examples of the XOR function. Thus the mean square error for epoch n is computed as follows,

$$E(n) = \frac{1}{4}\sum_{i=1}^{4}(d_i(n) - y_i(n))^2$$

where $y_i(n)$ is the network output for input i at epoch n and $d_i(n)$ is the desired or correct output for input i.

Along with your code, turn in a one-page description of your results that includes the parameters of your algorithm (learning rate, etc.), a graph that shows the mean square error as a function of the number of epochs of training, and the learned weights.

Extra Credit (up to 25 points): Flower classification using backpropagation

For extra credit, generalize your implementation of backpropagation from part II so that it can be used as a general supervised learning algorithm. In this case, the number of input and output nodes of your network, as well as the number of hidden nodes, should be parameters of your program that can be adjusted. One hidden layer should be sufficient.

In this extra credit part of the assignment, you will use the backpropagation algorithm to train a feedforward neural network to classify three types of Iris (Iris setosa, Iris versicolor, and Iris Virginica) based on four different attributes (sepal length, sepal width, petal length, and petal width). The training examples for this classification problem can be downloaded from MyCourses. For this problem, your network will have four input nodes and between one and three output nodes, depending on how you encode the output. (For example, you will use three output nodes if there is a distinct node for each type of Iris.) It is up to you to choose the number of hidden nodes that works best.

There are 150 examples in the Iris data file. You can use half for training, a quarter for tuning, and the remaining quarter for testing. This is called hold-out sampling. Even better, you can use the cross-validation method. For five points extra credit (included as part of the overall 25), implement momentum as part of your update rule.

Recall that an *epoch* is one pass through the training set. It is normal to take between 1000 and 2000 epochs to train your neural network. A learning curve plots the mean square error (or the classification error) against the training epoch.

For this part of the assignment, turn in your code and a one-page report describing how the learning algorithm performed on this test problem; a learning curve showing the effect of training in reducing the error; and a depiction of the network with the learned weights. In your report, discuss such questions as the effect of different learning rates (between 0.1 and 0.3 is typical), the effect of momentum (if you implemented it), and whether over-training occurred.