CS 6375 Machine Learning

Homework 3: Neural Networks [Total points: 100]

Due: Feb 25, 2015

Part I. Written problems. 50 points.

1. Gradient descent. [30 pts]
   A) We discussed in class using a sigmoid function or step function as the activation function in neural networks and derived weight update rules for such networks. In this homework, let us consider a different kind of computing unit for a neural network. Rather than a sigmoid function, it will use a Gaussian function. That is, given an input vector x (augmented with x_0=1) and a weight vector β, the output of the unit will be

   $$o = g(x \bullet \beta), \quad where\ g(z) = e^{-z^2}$$

   Derive the gradient descent training rule when using this function in a single layer perceptron with one output node.

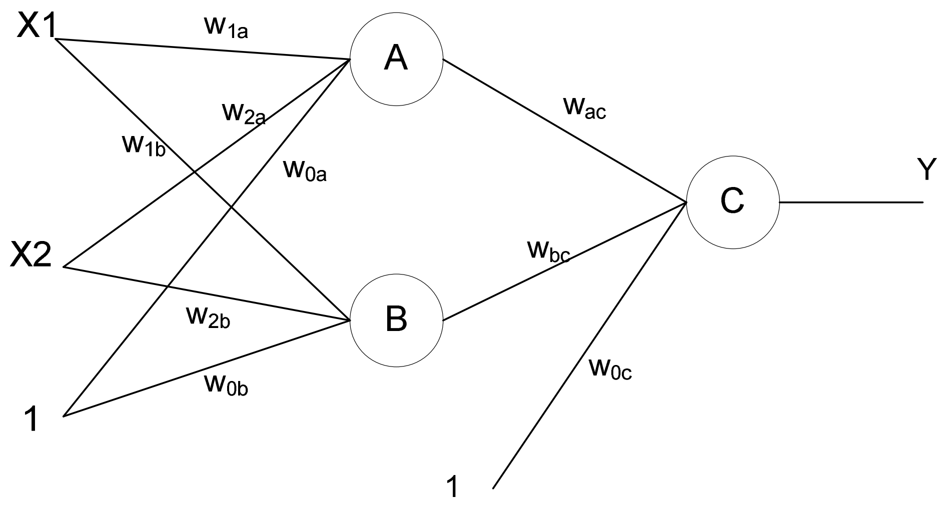
   B) In this problem we are going to use gradient descent algorithms to find parameters w_i for the following (regression) function.

   $$y = \sum_{i=0}^{D} w_i (x_i + x_i^{\ 2})$$

   Assume the update is done after reading one data sample with attributes x_i (i from 0 to D) and target value t. Find the weight update rule for w_i when using the squared error as the cost function.
   Note: this is not a perceptron or neural network like we discussed in the class. This question is only trying to help you better understand the general gradient descent method used to minimize some objective function.

2. Backpropagation [20 pts]

As we discussed in class, you need a network with hidden layers to implement XOR function. You designed a network like what's shown in the following (Note the bias term has a value of 1). Now use the backpropagation algorithm to find the weights.
Assume the initial weights are -0.01 for w_{1b} and w_{2a}, and 0.01 for all the others.
Show the weights after an example ((1,0), 1) is presented. Assume learning rate is 0.05. Sigmoid is used in all the nodes.



Part II. Programming. 50 pts.

In this assignment you will implement the gradient descent algorithm that we discussed in class to train **a single layer perceptron with sigmoid function** for binary classification tasks (i.e., each instance will have a class value of 0 or 1). You may use any programming language to implement the algorithm. To simplify the implementation, you may assume that all attributes are binary-valued (i.e., the only possible attribute values are 0 and 1) and that there are no missing values in the training or test data.

Sample training files (train*.dat) and test files (test*.dat) are available from the assignment page. In these files, only lines containing non-space characters are relevant. The first relevant line holds the attribute names. Each following relevant line defines a single example. Each column holds this example's value of the attribute named at the head of the column. The last column holds the class label for the examples.
These files are available at: http://www.hlt.utdallas.edu/~yangl/cs6375/homework/hw3/

Please initialize all the network weights to 0.

When applying the trained perceptron to a test instance, please use 0.5 as the classification threshold (i.e., classify the instance as 1 if the unit outputs a value that is at least 0.5; otherwise classify the instance as 0).

What to Do

a. Train the sigmoid unit on the training instances, using the given learning rate and number of training iterations. **Note** that (1) the algorithm will take one instance in each iteration and update the weights; (2) the number of training iterations can be greater than the number of training instances in the training set; (3) your learning algorithm should process the training instances in the same order as they appear in the training set; and (4) once you finish using all the training instances (that is called one epoch), you will start from the beginning of the training set again (if your training iteration is greater than the number of training instances).

b. Use the learned perceptron to classify the **training** instances. Print to stdout the accuracy of the perceptron. The accuracy should be computed as the percentage of examples that were correctly classified. For example, if 86 of 90 examples are classified correctly, then the accuracy of the perceptron would be 95.6%.
   Accuracy on training set (90 instances): 95.6%
c. Use the learned perceptron to classify the **test** instances. Print to stdout the accuracy of the perceptron.
   Accuracy on test set (10 instances): 60.0%

IMPORTANT:

- Your program should allow exactly four arguments to be specified in the command line invocation of your program: (1) a training file, (2) a test file, (3) a learning rate, and (4) the number of iterations to run the algorithm. Your program should take these four arguments in the same order as they are listed above. Any program that does not conform to the above specification will receive no credit.
- There should be no graphical user interface (GUI) of any kind.
- We may run your program on a new data set (with a different number of attributes) to test your code.

Submission instruction:

Programming part:
Submit via **eLearning** (i) your source code, and (ii) a README file that contains instructions for **compiling** and **running** your program (as well as the platform (Windows/Linux/Solaris) on which you developed your program). Again, you will receive **zero credit** for your program if (1) we cannot figure out how to run your program from your README file or (2) your program takes more than four input arguments.

Written part:
Please use a separate file for the written problems and submit it either via eLearning or give the hardcopy to the instructor or the TA.