CS 7840 - Soft Computing, Summer 2016
Assignment Two: Multi-Layer Perceptrons
Version 1.0 – Released on Feb 08, 2017

Objectives and Goals

This assignment is designed to help you become familiar with simple multilayer perceptron learning using the back propagation of error technique.

Deliverables

Please prepare and turn in answers to each of the provided discussion questions. You must turn in your materials via the Pilot on-line learning system. Your answers and supporting materials for this homework assignment must be turned in by 11:59 PM on the due date.

ALL of your deliverables for this homework assignment should be placed in a single archive file. You may use either of the following:

zip	http://en.wikipedia.org/wiki/ZIP_%28file_format%29),
gzip	http://en.wikipedia.org/wiki/Gzip

Your archive file, which you will turn in using Pilot, should unpack into a single folder called "HW2_yourlastname" where "yourlastname" is replaced with your last name. Inside that folder, you should have placed various other files and folders as described in this document.

Your written answers to all questions, including supporting graphics and data tables, should be encoded as a SINGLE PDF file. This file should have the name yourlastname_HW2.pdf. In addition, the content of the document itself should clearly state your name and the title of the assignment (Assignment Two: Multilayer Perceptrons).

For the programming assignment, you should have one folder called SOURCE_CODE and contain ALL source code, data files, and written explanations (if any) about your work related to the programming assignment.

Note that sample code is available to get you started. That code is available for download from Pilot.

Discussion Questions

Your answers to these questions should be placed in the YOURNAME_HW2.pdf file that appears inside the zip archive you turn in for this assignment.

1. Suppose one is given a three-layer network that processes a pair of inputs and produces a single output. Assume the first (hidden) layer uses an activation function of the form:

   $$\phi(v) = 1/(1+e^{-v})$$

   and the second (hidden) layer use a activation function of the form:

   $$\phi(v) = (1-e^{-2v})/(1+e^{-2v})$$

   and the third (output) layer uses a linear activation function of the form:

   $$\phi(v) = v$$

If the initial weights and biases for this network are:

Layer 1 (hidden)	Layer 2 (hidden)	Layer 3 (output)
$\begin{bmatrix} b_1 & b_2 \\ w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = \begin{bmatrix} 0.1 & -0.1 \\ 0.1 & 0.3 \\ -1.0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} b_1 & b_2 \\ w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = \begin{bmatrix} -0.5 & 0.1 \\ 0.1 & -0.1 \\ -0.2 & 0.1 \end{bmatrix}$	$\begin{bmatrix} b_1 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.5 \end{bmatrix}$

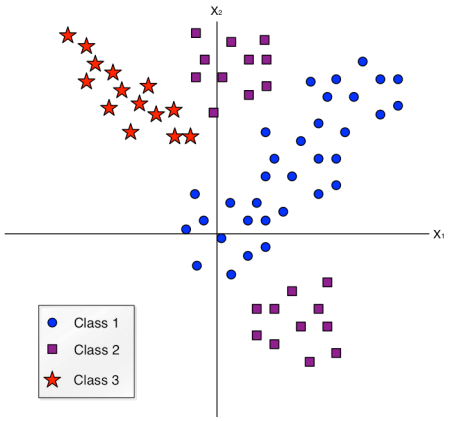
Perform a single iteration of the back propagation algorithm with a learning rate of 0.01 for the training sample composed of the input/output mapping of: $(x_1 = 1,\ x_2 = 1,\ y_d = 1)$. Write out all the steps taken in the iteration with appropriate calculations. BE NEAT AND PROVIDE CLEAR EXPLANATIONS OF EACH STEP to demonstrate your understanding back propagation at the level of individual arithmetic computations.

2. When using the back prop / GDR algorithm, why it is critical that outputs of neurons not be allowed to saturate to their maximum high or low values during training. Explain why and give one strategy that one may use to prevent that outcome. Explain how your strategy accomplishes that.

3. In your own words, what a momentum term and why is it used? What problem is it trying to solve and why does the use of momentum terms solve it?

4. Inputs to a multi-layer perceptron are often "pre-conditioned" by normalizing all individual components of $\vec{X}$ to lie in the range of $[-1.0\,..\,1.0]$ before presenting them to the neural network for either training or generation of outputs. Provide two distinct reasons why this is considered a good practice. Explain what can go wrong if one does not do this and how those problems are prevented by doing it. BONUS: Provide a "downside" to preconditioning inputs as indicated. Is there something one could do to avert the "downside" you have identified?

5. Design a multi-layer perceptron of minimal size (minimum number of neurons) capable of properly categorizing data from the three classes shown in the training data in the graph to the right. Be sure to explain exactly WHY you chose to have a certain number of neurons in the output layer and a certain number of neurons in the hidden layer and why you consider those choices to constitute a "minimum sized" network. Also provide weights and biases for you network that successfully classify the patterns shown to the right. Demonstrate that your specific choices of weights and biases actually solve the classification problem given.



Programming Assignment

All of your source code for this problem should and any other deliverables you want us to see (text files, PDFs of images or graphs, etc.) should be placed in a folder called SOURCE_CODE. You should include along with that source code directions for compiling and running your program. You should also include a PDF file called "testing.pdf" that explains how you tested your code and why you believe if satisfies the requirements of the assignment. You can and should include screenshots and/or calculations that prove your code worked as intended. The instructor and/or graders will also compile and test your code.

Complete each of the following exercises, putting the source code and other requested files in the SOURCE_CODE directory inside the archive you turn in.

1) From Pilot, download the "hw2_backprop_examples_2017.zip" archive. Unpack it and examine the contents. You will find one C code program that solves a simple pattern recognition problem similar to that given in homework problem #5 from the Discussion Questions section of the homework. You will also find text files containing the raw data points for a "Class_1", "Class_2", and "Class_3". These are the same points that are used for training in the classification sample code.

2) Create a program to train a classifier to assign patterns to the Class_1, Class_2, or Class_3 as defined in the text files in the backprop_examples archive. You may either create your program from scratch in any language you like or you may expand the provided sample code so that it satisfies the following minimum requirements:

   - Your program should be able to properly compute outputs for and train a two layer perceptron with any number of input units and any number of hidden and output layer neurons. *The sample code already does this*.

   - Your program should be able to print out the weights and biases of any network it can represent. *The sample code already does this*.

   - Your program should implement *epoch training* (I.E. weights are updated based on the sum of all corrections for ALL patterns in the training set. *The sample code **does not** already do this*.

   - Your program should continue training ONLY until all patterns in the training set are correctly categorized. *The sample code **does not** already do this*. It should also keep track of how many training epochs occurred from the initial randomization of weights until the time at which all training patterns were correctly categorized. At the end of training, your program should print out the number of epochs it took to achieve zero error on the training data.

   - For this assignment, you should randomize all weights and biases to small values prior to training (*The sample code already does this*). You can, for this assignment, make arbitrary choices for the range of random weights so long as you verify your choices do not prevent proper training. For this assignment, you need not normalize your inputs into the range of [-1 .. 1], again, so long as you verify that for the data given, training is possible. We will discuss input normalization and weight initialization heuristics later in the course.

   Once your code is completed, do the following:

   a) Train a neural network with one hidden layer of four neurons and one output layer of two neurons to properly classify Class_1, Class_2, and Class_3 data into three separate labeled categories. Verify that all patterns are classified correctly. Repeat training at least 20 times and for each training run, record the number of epochs required to achieve zero error on the training set. Compute the average time, in epochs, required to hit zero error (defined as all patterns are properly classified).

   b) Repeat (a) above with the following architecture: 2 input units, 8 hidden layer neurons, and 2 output neurons. Train this network the same number of times you trained the network for item (a) above. Compute the average time, in epochs, required to hit zero error. Is this time greater or lesser than the

time computed in (a)

c) Repeat (a) above with the following architecture: 2 input units, 2 hidden layer neurons, and 2 output neurons. Train this network the same number of times you trained the network for item (a) above. Compute the average time, in epochs, required to hit zero error. Is this time greater or lesser than the time computed in (a)

d) Report the average training times for (a), (b), and (c) in a text file include with your source code.

Double Dog Dare Question (Optional)

The file double_dog_dare_data.zip unpacks into an ASCII file of labeled class data. The first field of each line is 1.000 or -1.000, which is the label for one of two classes. The rest of the fields in a line are processed sensor readings associated with the class. There are 100,000 labeled class vectors in the file.

Your double dog dare mission is to create a neural network classifier that can properly classify input vectors into one of the two classes. You should verify performance, if you attempt this, with data that was never seen during training.

This problem is horrifically difficult, I'm not even entirely sure how I'd go about doing it. Nonetheless, thinking about HOW you'd do it is a good exercise and set up for homework #3.

Anyone who *can* solve the double dog dare problem with a classification accuracy of 95% or greater will win a prize. That being said, don't waste time on this until the rest of the homework is done and turned in.