EECE 5136/6036: Intelligent Systems
Homework 2: Given 10/2/22; Due 10/16/22

This homework with begin to introduce you to real-world data, though still on a very small scale. You will use one of the most widely used data sets for evaluating machine learning applications in the image analysis area: the MNIST dataset (http://yann.lecun.com/exdb/mnist/), which provides images of handwritten digits and letters. In this homework, you will use the numbers subset from this dataset, and only a subset of this in a very simple context. (If you are interested in the larger dataset with handwritten characters, look at https://www.nist.gov/itl/products-and-services/emnist-dataset).

Problem 1. (200 points) The goal in this problem is to train a single threshold neuron shown in Slide 22 or Lecture 2 to discriminate between two handwritten digits. The training will be done using the simple reinforcement paradigm illustrated in Slide 40 of lecture 2.

Two data files are included in this homework:

- **Image Data File:** *MNISTnumImages5000.txt* is a text file that has data for 5,000 digits, each a grayscale image of size 28 $\times$ 28 pixels (i.e., 784 pixels each). Each row of the data file has 784 values representing the intensities of the image for one digit between 0 and 9. The first hundred images are shown in the included file *first100.jpg*. There are 500 samples of each digit. To turn a line of data into an image, reshape the 1×784 vector into a 28×28 matrix and plot as a heatmap image.
- **Label Data File:** *MNISTnumLabels5000.txt* is a text file with one integer in each row, indicating the correct label of the image in the corresponding row in the image data file. Thus, the first entry '7' indicates that the first row of the image data file has data for a handwritten number 7.

You need to do the following:

1. Using the label files, separate out the images for 0, 1, 7 and 9 into separate files. This will give you four image files of about 500 images each. Take 400 points each from the 0 and 1 files, put them in a single file of 800 points, and randomize the order of the images (mix them up so not all 0s or 1s are together). Make sure you keep track of which row in the randomized file is 0 and which is 1. Now you have your *training set*. Take the remaining 100 points from the 0 and 1 files, and make another file with these 200 images, again keeping track of which ones are 0 or 1 (it is not necessary to randomize the order here.) This is your *test set*. Also select 100 points each from all the other sets (2 through 9), and combine them into a single set, making sure to keep track of which inputs are which digit. This is your *challenge set*. Again, it is not necessary to randomize the order here.

   You will use the full MNIST dataset in some future homeworks, and this homework will familiarize you with this dataset.

2. Write a program to simulate the binary threshold neuron shown in Slide 22 of Lecture 2. The neuron has 784 inputs, x_1 through x_{784}, one for each pixel of your images. Thus, the first pixel (upper left corner) of the image is input x_1 and the last pixel (lower right corner) is x_{784}. Each input line has a corresponding weight, w_j, where $j = 1, 2, ..., 784$. The weights are all initialized to random values between 0 and 0.5. Save the initial set of weights.

   Images are input to the neuron one at a time, so the *net input* to the neuron at time step t is:

$$s(t) = \sum_{j=1}^{784} w_j x_j(t) \tag{1}$$

   where $x_j(t)$ is the jth pixel value of the current input image.

The neuron also gets a *teaching input*, $z(t)$, which can be either 0 or 1. During the training phase, the neuron's output $y(t) = z(t)$, i.e., it is always the same as the teaching input. Once training is finished, the teaching input is removed, and the neuron's output $y(t)$ depends on its net input, $s(t)$ as given below:

$$y(t) = \begin{cases} 1 & \text{if } s(t) > \theta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where θ is a threshold that will be set after training.

3. Train the network through several *epochs*, where an epoch is one pass through the whole training set. At each training step, one of the training images is input to the neuron. If the image is a 0, $z(t)$ is set to 0. If the image is a 1, $z(t)$ is set to 1. Thus, the neuron fires only for 1s. At each step, once the neuron's output has been determined, use the post-synaptically gated Hebb rule defined in Slide 42 of Lecture 2 to adjust the weights of the neuron. Thus, each weight w_j will change as follows at time step t:

$$w_j(t) = w_j(t-1) + \eta y(t)[x_j(t) - w_j(t-1)] \tag{3}$$

Make sure that η is very small (around 0.01 or less). Train for at least 40 epochs (this depends on your η - if it is very small, you need more epochs, but 40 should be ok with $\eta = 0.01$).

4. After training, remove the teaching input. Vary θ values from 0 to 40 in steps of 1. For each value of θ, present all the images in the *test set* to the trained neuron and get the outputs. From these outputs, identify the true positives (1s identified as 1), true negatives (0s identified as 0), false positives (zeros identified as 1), and false negatives (1s identified as 0). From these, calculate the precision, recall, and the F1 score at each value of θ.

5. Plot the precision, recall, and F1 against the values of θ – all three plots in the same graph with θ on the x-axis. Also plot the ROC curve (true positives on y-axis vs. false positives on x-axis) as a separate plot. From the ROC curve, estimate the best value of θ, report it, and explain (in 3 lines max) why you chose that value.

6. Take the 784 weights of the neuron before training and after training, rearrange each set as a 28×28 matrix, and plot them side by side as images or heatmaps.

7. Using the optimal threshold, present the inputs from the challenge set and see which points are classified as 1 and which ones as 0. Using this data, calculate how many inputs datapoints of each type are classified as 1 and how many as 0. For example, how many 7s out of 400 are classified as 0 and how many as 1, and so on.

   Show these values in a 2×8 matrix as a table or figure. The rows here correspond to 0 and 1 and the columns to the 8 digits from 2 to 9.

Write a brief report providing the following information. Each item required below should be placed in a separate section with the heading given at the beginning of the item.:

- **System Specification:** State the number of epoch and the η value you used, and why you made those choices. You may need to try several values before finalizing them (5 lines, single spaced, 12 point type).
- **Results:** Plot all the figures asked for in the itemized problem statement, making sure that all graphs are properly labeled, axes are labeled, and each figure has a number and caption (just the plots and 3 lines max explaining the choice of optimal θ).
- **Analysis of Results:** Briefly interpret and discuss your results (8 lines, single spaced, 12 point type). In particular, analyze the results on the challenge set and discuss why some digits are more likely to be classified as 0 or 1.

Problem 2. (200 points) Repeat Problem 1 with the same training, test and challenge sets, but instead of using the reinforced neurons from Problem 1, use a perceptron. A 1 image should produce a 1 output and a 0 image a 0 output.

Your perceptron will have 784 inputs from the image and one bias input that is always set to 1. It will have 785 weight (784 from the input image and one from the bias). As in Problem 1, you will randomly initialize the weights and train by sequentially presenting each image in the training set, adjusting the weights at each step. As discussed in class for the perceptron, at each step, you will produce the actual output, get its error from the desired output, and use that to adjust all the weights with a small learning rate.

You will need to go through several epochs as in Problem 1. How many epochs you run is up to you, but you should try to get the error to converge to as low a value as possible.

You should calculate the following and plot graphs as indicated below:

1. During training, after every epoch, calculate the *error fraction* of the network on the training and test sets (train only on the training set, of course). The error fraction is the fraction of input images for which the perceptron produces incorrect output. Thus, if, at step t, the perceptron is incorrect on 20 out of 400 1s and 60 out of 400 0s, the error fraction is (20+60)/800 = 0.1. You will do a similar calculation for the test set. As a result, you will get two error fraction values at each epoch. After you are done training, plot the training error and the test error versus epoch on the same graph. Thus, the x-axis of the graph will be epoch number and the y axis will be the error. Make sure that the two curves are in different colors or differentiated in some other way.
2. After initialization but before training, calculate the precision, recall, and F1 score of the perceptron. Calculate these scores again after training is completed. Plot these three scores before and after training as a paired bar plot: three pairs of bars, one each for precision, recall and F1, with the left bar in the pair the score before training and the right bar the score after training.
3. As in Problem 1, present the images in the challenge set to the perceptron after training and obtain the 2×8 table as in the previous homework. Show that table.
4. As in Problem 1, show the initial weights and the final weights of the perceptron (not including the bias) as 28×28 images.

Note that there is no ROC curve in this case because you are not varying the threshold as in Problem 1. Here, the algorithm has trained the best threshold (the bias weight). You need to plot only the graphs and tables listed above.

Write a brief report providing the following information. Each item required below should be placed in a separate section with the heading given at the beginning of the item.:

- **System Specification:** State the number of epochs and the η value you used, and why you made those choices. You may need to try several values before finalizing them (8 lines, single spaced, 12 point type).
- **Results:** Plot all the figures asked for in the itemized problem statement, making sure that all graphs are properly labeled, axes are labeled, and each figure has a number and caption.
- **Analysis of Results:** Compare the precision, recall and F1 score obtained with the perceptron after training with the scores you got in Problem 1 at the best value of θ. In 10 lines or less, discuss the similarities and differences in the two cases, and your thoughts on the reasons underlying the differences (if any). Indicate whether one system able to handle the challenge set better than the other, in what way, and why (in your opinion)? Which method would you prefer?

For this homework, you will probably need to do some trial and error in terms of weight initialization, learning rate and duration of training (number of epochs) before finalizing those values. You should only show the results for your final version, but in item 1 of the report, you should state how you chose the final values of these things.

Report Instructions:

Please follow the instructions given for previous homework.

Submission Instructions:

You should submit your report on-line through Canvas. Your submission will have three documents: 1) A report as described above; 2) A file (or .zip file) with all your source code; and 3) A brief README file with instructions on how to compile (if needed) and run the code. If Canvas does not let you submit a file with the type postfix (.py, .m, etc.) of a source code file, try changing the postfix manually to .docx or .txt and submit it. State in the README file what the postfix should be changed to in order to run the code. **Note that a .zip file can be used ONLY for the code. Your report and README files must be in Word or PDF.**

Grading:

Points will be awarded for correctness of the results, proper plots, and the clarity of description and conclusions.

You may consult your colleagues for ideas, but *please write your own programs and your own text*. Both the text of the reports and the code will be checked randomly for similarity with that of other students. *Any text or code found to be copied will lead to an automatic zero on the entire homework for all students involved*. Repeat offenses in the future will incur more severe consequences.

If you cannot access the data, please send me mail at. Ali.Minai@uc.edu.