EECE 5136/6036: Intelligent Systems
Homework 2: Given 10/4/23; Due 10/17/23

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).

You are not allowed to use ChatGPT or any other generative AI, or any pre-implemented library version of the perceptron for this homework.

Problem 1. (200 points) The goal in this problem is to train a single perceptrom to discriminate between two handwritten digits.

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$\times$784 vector into a 28$\times$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, take 400 points each from the 0 and 1 digits, 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 0 and 1 points 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 digits (2 through 9), and combine them into a single dataset, 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.
2. Write a program to simulate the perceptron. 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 bias input, x_0 is always set to 1 and also has a trainable weight, w_0. 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:

   $$s(t) = \sum_{j=0}^{784} w_j x_j(t) \tag{1}$$

   where $x_j(t)$ is the jth pixel value of the current input image. The output, $\hat{y}(t)$, is 1 if $s(t) > 0$, else it is 0.
3. After initialization of the weights but before training, present all the images in the *test set* to the untrained 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. Also calculate the *error fraction* of the network on the training and test sets. 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 40 out of 100 1s and 60 out of 100 0s, the error fraction is (40+60)/200 = 0.5. All these values will generally be quite poor because the system has not been trained yet.

4. Train the perceptron 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 and the weights are changed according to the percetron learning rule described in Lecture 5. Make sure that η is very small. Train until your error fraction on the training set becomes small enough.

5. During training, after every epoch, calculate the error fraction on the training set. After you are done training, plot the training error fraction versus epoch as a time-series, starting with the initial value. Thus, the x-axis of the graph will be epoch number and the y-axis will be the training error fraction.

6. After training, calculate the error fraction, precision, recall, and F1 score on the test set. Plot these four values along with the corresponding values before training as a bar plot. Your plot will have four pairs of bars, each pair showing is metric's value before and after training. Keep in mind that these are all on the test set, not the training set.

7. Take your trained bias weight, w_0, and choose a range of 20 values around this - 10 lower than the learned bias weight and 10 higher, symmetrically distributed, with the lowest value about 20% lower than the trained weight and the highest value about 20% higher. Keeping all other weights the same, set the bias weight w_0 to each of these 20 values, and in each case, get the error fraction, precision, recall, and F1 score on the test set. Plot the four metrics against the 21 values of w_0 (the 20 new values and the trained value in the middle. All four plots should be on the same graph with w_0 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 w_0, and report it. Is the trained value the best? If not, how much better did some other value do?

8. Take the 784 weights of the neuron (excluding the bias weight) before training and after training, rearrange each set as a 28×28 matrix, and plot them side by side as images or heatmaps.

9. Using the trained weights and bias, present the inputs from the challenge set to the neuron 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 100 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.

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 for this problem. 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.

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 (100 Points)

Repeat Problem 1, but this time use 0 and 9 as the two classes to be discriminated (0 = false, 9 = true), and use the other 8 digits as your challenge set. Train for the same number of epochs and the same learning rate you used in Problem 1. You should use a new set of random initial weights, but the method for choosing them must be the same as for Problem 1. Draw all the same graphs as in Problem 1.

Discuss briefly (half-page max) how the performance of the perceptron was on this task compared to the one in Problem 1 (on both the test and challenge sets), what might explain the difference (if any).

Report Instructions:

Please follow the instructions given for previous homeworks.

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.