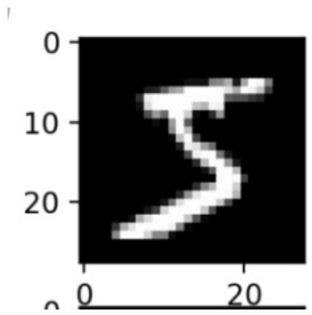
CS 480 Artificial Intelligence Project 4 Fall 2022

Due: Nov 15, 2022

The project will be done in pairs. Each pair will submit just one program. Make sure to include both names in all the source files.

This project involves building a linear classifier that uses the perceptron training algorithm to identify the hand-written digits from the images of the MNIST dataset such as the one shown below.

The input to the perceptron will be ten features extracted from the images: (1) the density, (2) degree of symmetry which will be defined below, $(3-7)$ maximum and average number of horizontal and vertical intersections and **three** additional features of your choice.



To implement the project, please follow the steps below:

Step 1: Write a function to extract the ten features from a vector of length 28×28 containing values between 0 and 255 representing a 28×28 gray-scale image.

Step 2: You are given ten training set files (train0.csv, ..., train9.csv) each containing the pixel values of the images of letters 0 through 9. Each file contains 1000 images. Extract the features from all the 10000 images and create a single training set TRAIN with 10 columns and 10000 rows. Add two extra columns (-1 for the threshold weight as column 11 and the class label as column 12. Randomly permute the rows of TRAIN.

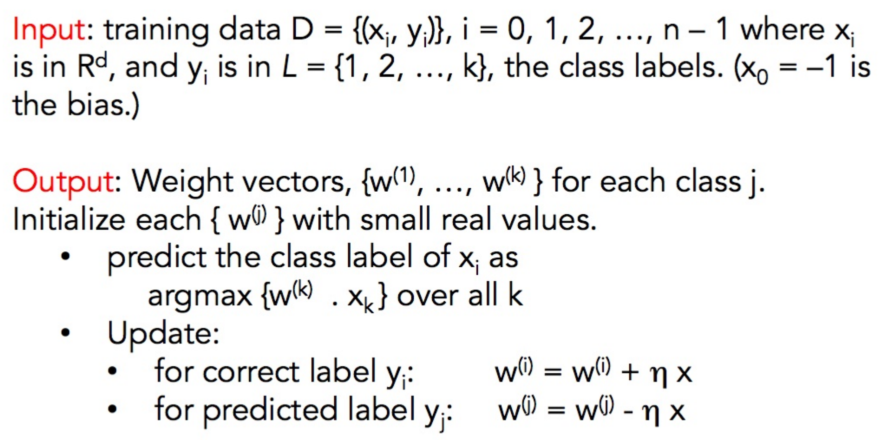
Step 3: Use the algorithm shown below for multiclass linear classifier by initializing 10 weight vectors $\mathbf{w_j} = (w_j^{(0)}, w_1^{(j)}, \cdots, w_{10}^{(j)})$, $j = 0, 1, \cdots, 9$ randomly using small constants in the interval (-0.05, 0.05) and adjust the weights of the vectors $\mathbf{w_j}$ as described. Each epoch will involve training on each of the 10000 instance once.

Step 4: At the end of an epoch, use the validation data (valid0.csv, ... , valid9.csv) to evaluate the total number of errors made (over all the 2500 validation data points).

Repeat the above Steps 100 times (i.e., for 100 epochs), and keep track of the weights associated with the lowest error found after each epoch. At the end of 100 epochs, report the ten weight vectors and the minimum error found.

The final step is to take a file named test.csv (that contains 100 images - ten for each digit), extract the features and use the optimal weight found above to identify the digit. The output from this step will be a sequence of 100 digits.

Features: Density is defined as the average gray scale value of all the pixels in the image and is thus a real number between 0 and 255. Measure of symmetry is defined as the average gray scale of the image obtained by the bitwise XOR ($\oplus$) of each pixel with its corresponding vertically reflected image. (Thus if I is the image, let I' be the image whose j-th column is the $(28 - j)$-th column of I. Then, the measure of symmetry is the density of $I \oplus I'$.) The number of vertical intersections is defined as follows: First turn the image into black and white image by assigning a color 0 (1) for gray scale values above (below) 128. Consider the j-th column as a string of 0's and 1's and count the number of changes from 0 to 1 or 1 to 0. For example, the number of changes for string 11011001 is 4. Average this over all columns to get the average number of vertical intersections, and maximum over all columns will give the maximum number of vertical intersections. Vertical measures are defined in a similar way based on rows.)



Input: training data D = $\{(x_i, y_i)\}$, i = 0, 1, 2, …, n – 1 where x_i is in R^d, and y_i is in $L = \{1, 2, …, k\}$, the class labels. ($x_0 = -1$ is the bias.)

Output: Weight vectors, $\{w^{(1)}, …, w^{(k)}\}$ for each class j.
Initialize each $\{ w^{(j)} \}$ with small real values.

- predict the class label of x_i as
  argmax $\{w^{(k)} . x_k\}$ over all k
- Update:
  - for correct label y_i: $w^{(i)} = w^{(i)} + \eta$ x
  - for predicted label y_j: $w^{(j)} = w^{(j)} - \eta$ x

What should be submitted?

When your program runs, it uses the training data and validation data to complete the training phase and prints the ten optimal weight vectors $\mathbf{v_j}$, $j =), 1, \cdots, 9$ and reports the total number of errors and the error percentage. Then, it outputs a sequence of 100 digits by classifying the images from the file test.csv (which is assumed to exist in the current directory.) Create your own test file, and the run the program and include the output produced. (Your program will also be tested on a different test file.)