CS 480 Artificial Intelligence Project 4 Spring 2020

Due: April 15, 2020

The projects will be done in teams of up to three. Each group will submit just one program. Include the names of all members in the zip file.

This project has two parts. The first part involves building a classifier that uses the perceptron training algorithm to differentiate between two specific digits (7 and 9) from the MNIST dataset. The input to the perceptron will be the following seven features extracted from the images of digits 7 and 9: (1) the density, (2) degree of symmetry which will be defined below, $(3-7)$ maximum and average number of horizontal and vertical intersections. Divide the training sets data7 and data9 into training and test sets with 80% and 20% validation data. Then extract the features from the sets and merge the training sets for data7 and data9 into a single set TRAIN. Run the perceptron training algorithm for a maximum of 1000 epochs on TRAIN. At the end of each epoch, determine the success rate on the validation data (which has 20 percent of letter 7 and 20 percent of letter 9). Keep track of the weights associated with the perceptron with the least error rate on the validation data. Choose $\eta = 0.1$ and initialize the weights to random real numbers between -0.1 and 0.1. Keep track of the weight associated with the minimum error on validation data and output this weight. After the weights are determined using the above algorithm (called the pocket algorithm), your program takes a test file and output a vector of 7's and 9's that represent your classifier's class labels on the test data. The output should be a text file containing a sequence of 7's and 9's separated by at least one blank space between successive 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.)

The second part of the project is to implement the multiclass perceptron training algorithm (shown below) to identify the handwritten digit using MNIST data set. In this part, we will use all the ten digits with the same number of training and test data as in part one. The details of implementation are similar to part 1. (Choose η and the initial weights as in part 1.)

What should be submitted?

When your program runs, it should first provide a solution to part 1: it should take as input four files - two training data sets for digits 7 and 9, and two validation data sets for digits 7 and 9. Assume that the names of the files are train7.txt, train9.txt, valid7.txt and valid9.txt. It should train the perceptron for 1000 epochs. After each epoch, measure the error on the test data (test7 and test9). It should utput the weights associated with the perceptron that has the

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 $L = \{1, 2, \ldots, k\}$, the class labels. ($x_0 = -1$ is the bias.)

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

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

least validation error, and output this minimum error fraction (as a fraction between 0 and 1.) Then it should read the file test.txt and classify the test data with a sequence of 7's and 9's as output.

For part 2, your program will go through the same steps as above, but use 20 files train0.txt, train1.txt, ... , train9.txt, test0.txt, test1.txt, test2.txt, ..., test9.txt. The criterion to determine the error associated with the current weight vector on validation data, use the sigmoid function to determine the log likelihood $ll(w) = \Sigma_i log P(y^{(i)}|x^{(i)}; w)$. The final output is again based on test_final.txt and it will be a string of digits $0, 1, \cdots, 9$.