Perceptron learning rule: selecting the input space

In this homework, we will consider handwritten digits and try to classify them.

This training dataset is derived from the original MNIST database available at http://yann.lecun.com/exdb/mnist/

We have processed the database and provide a separate training data file for each class 0 to 9. Use the following links to download them. Right click and save target in a desired folder.

data0 data1 data2 data3 data4 data5 data6 data7 data8 data9

File format:
Each file has 1000 training examples. Each training example is of size 28x28 pixels. The pixels are stored as unsigned chars (1 byte) and take values from 0 to 255. The first 28x28 bytes of the file correspond to the first training example, the next 28x28 bytes correspond to the next example and so on.

To read the files, here is what to do. In Matlab, use the following code

fid=fopen('data8','r');-- open the file corresponding to digit 8
[t1,N]=fread(fid,[28 28],'uchar'); -- read in the first training example and store it in a 28x28 size matrix t1
[t2,N]=fread(fid,[28 28],uchar); -- read the second example into t2
and so on
To display the image use imshow(t1) or imagesc(t1)

Note: Make sure you are reading the files correctly. Check by displaying the first few and the last few images in each class.

Homework
We again want to use the perceptron learning rule in order to classify a given image, but now we must create our own 'features', based on the given images. Two of the easiest numbers to differentiate are the 0s and 1s, so focus on these two groups.

1) First we want to create an input of values using all of the given data. We need to derive features from the data that could be used to easily detect a 0 or a 1. Three of many such features include 1) the total average pixels located at the center of the image 2) the total average pixels over the entire location and 3) the symmetry of the image.

   Calculate these three variables for each of the 1000 images of 0 and 1, and save these features as the input data.

$$x = [value1_1\ value1_0\,; value2_1\ value2_0; value3_1\ value3_0]$$

   *Note, the calculations are more manageable if you go through and convert each of the pixels in the 28x28 matrix to a binary value first.

   You should end up with a 2000 x 3 input matrix, with the first 1000 features corresponding to all of the 1s and the second 1000 rows corresponding to the three features for all of the given 0s.

Plot each of these three features, in order to view the separation between the values for the 0s and the 1s.

Make sure to also save the true output for each of these 1000 features, in order to compare the true classification, y as a 0 or 1 and the predicted classification, $\hat{y}$ based on the input data.

2) Starting from a 3x1 vector w=[0 0 0] , use the following to predict the value of the handwritten image:

$$\hat{y}_i = sign[w^T x_i]$$

Count the number of wrong predictions for 200 randomly selected images. For each image that your prediction is wrong, use the following learning rule to update your weight vector:

$$w \leftarrow w + y_i x_i$$

3) Repeat this training/testing for 500 times, and plot the number of wrong predictions per each step.