ENEL 525 – Lab 3 – LMS Algorithm

25 October (session I) & 1 November (session II), 2023

Objective: Implement the least mean square error (LMS) learning algorithm for a single layer network with multiple neurons and the linear transfer function to solve a classification task and a character recognition task.

Part 1: Training a neural network using the LMS Algorithm

1. Consider the following classification problem. The four classes are:

$$class_1: \left\{\boldsymbol{p}_1 = \begin{bmatrix}1\\1\end{bmatrix}, \boldsymbol{p}_2 = \begin{bmatrix}1\\2\end{bmatrix}\right\}$$

$$class_2: \left\{\boldsymbol{p}_3 = \begin{bmatrix}2\\-1\end{bmatrix}, \boldsymbol{p}_4 = \begin{bmatrix}2\\0\end{bmatrix}\right\}$$

$$class_3: \left\{\boldsymbol{p}_5 = \begin{bmatrix}-1\\2\end{bmatrix}, \boldsymbol{p}_6 = \begin{bmatrix}-2\\1\end{bmatrix}\right\}$$

$$class_4: \left\{\boldsymbol{p}_7 = \begin{bmatrix}-1\\-1\end{bmatrix}, \boldsymbol{p}_8 = \begin{bmatrix}-2\\-2\end{bmatrix}\right\}$$

Design a single layer neural network with a linear activation function and implement the LMS algorithm to solve this problem. Error threshold and learning rate will need to be determined experimentally.

2. It is unlikely that the mean squared error (MSE) will be reduced exactly to 0 in a reasonable amount of time. To prevent iterating forever, use a small valued (constant) error threshold as the condition to stop iterating when the MSE is reduced below the error threshold. One iteration involves a pass over each of the input patterns. Note that the MSE mentioned here is the MSE of all the patterns in an iteration, not just an individual pattern.

3. The MSE can be calculated using the mse() function given below. Store the scalar MSE values from each iteration in a vector for plotting later (length will be equal to the number of iterations).

```
def mse(array):
    return np.mean(array.flatten() ** 2)
```

4. Plot the learning error curve. Use `matplotlib.pyplot.semilogy()` for this plot.

5. Apply the trained network to classify the input vectors. Create a 4x8 table of squared error values between each target vector and the output obtained for each input vector (compare to all targets, not just the target for the given input). This can be done using mse(err_{ij}) where $err_{ij} = target_i - output_j$. This is an error measure, so the correct class will be indicated by error values closest to zero.

Part 2: Character Recognition

A set of 26 lower-case alphabets images will be provided in the lab. Implement the LMS learning algorithm to train a single layer neural network. This time train the network for autoassociation using a linear activation function. Observe the effect of MSE threshold by plotting the learning error curve. Apply the trained network to recognize the 26 characters and analyze the testing performance quantitatively and visually. Do this by creating a correlation table using `scipy.stats.pearsonr()` as in lab 2, and display a few of the output characters to visually confirm that the correct output is obtained.

Hint:

1. Load and preprocess all 26 images.

a. Reading an image: Save the given images to the local directory. Use the function `Image.open()` to read an image to a matrix.

b. Convert each matrix into a column vector and append each image as a new column to create the input matrix.

c. Normalize the matrix using `sklearn.preprocessing.normalize()`.

2. Apply all 26 training characters and the LMS learning rule to update the weight matrix and bias until convergence. Error threshold and learning rate will need to be determined experimentally.

3. As in part 1, plot the learning curve using `matplotlib.pyplot.semilogy()`.

4. Apply the trained network to recognize the 26 training characters. Evaluate the testing errors quantitatively by calculating the correlation coefficient between the network output and desired target output using `scipy.stats.pearsonr()`.

5. Finally, test the performance of the trained network visually and display the image using `matplotlib.pyplot.imshow()`.

Marking:

Show the following to the TA during the lab period:

Part 1:

• Show the learning curve plot (i.e. MSE vs iteration number).

• Create a table of squared error values between the possible targets and the outputs given each of the input patterns (will be a 4x8 table).

Part 2:

• Show the learning curve plot (i.e. MSE vs iteration number).

• Display a few of the output characters to visually confirm the network correctly recalls the input characters.

• Create a table of correlation coefficients between the possible target characters and the network outputs (will be a 26x26 table).

The TA may apply different/additional inputs to your code.