THE UNIVERSITY OF CALGARY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ENEL 525 Machine Learning for Engineers

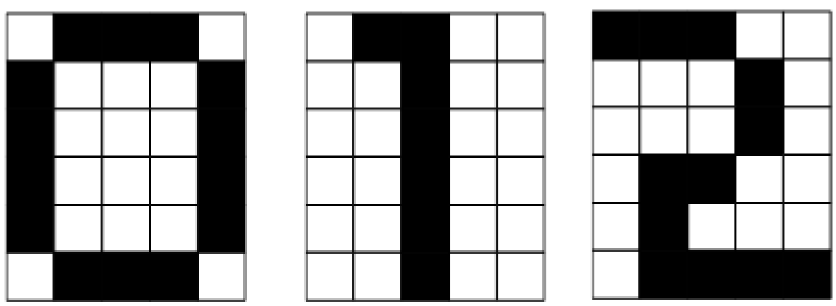
Lab 2 – Associative Memory

7 October (B02) & 10 October (B01), 2025

Objective: To perform face recognition using linear associator based on Hebbian learning rule and pseudo inverse rule.

Part 1:

1. Create a new .m file. A set of patterns are as follows:



2. Convert each pattern to a single row vector, in which black is denoted by -1 and white is represented by 1. Normalize them via the function sklearn.preprocessing.normalize(). For example, the first pattern is represented as **P**1 = [1 -1 -1 -1 -1 1 -1 1 1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 1 1 -1 1 -1 -1 -1 -1 1]^T.
3. Apply the Hebbian learning rule on the resulting vectors to create a weight matrix. Make the desired output vectors equal to the input vectors to form an autoassociative memory.
4. Randomly reverse 3 pixel values of each given pattern. Vectorize and normalize the noisy patterns.
5. Apply the trained network to recognize the noisy patterns. Reshape the output vectors into matrices (use function numpy.reshape()) and observe the recognition performance of the network (use function matplotlib.pyplot.imshow()). In addition, calculate the correlation coefficient between each network output and each input pattern and complete Table 1. The similarity/difference between two images is obtained by calculating correlation coefficient, which indicates the degree of linear relationship between two patterns. Correlation coefficient ranges between 0 and 1, in which 1 refers to the highest similarity while 0 denotes the weakest. In Python, the correlation coefficient can be computed by the function scipy.stats.pearsonr().

Table 1

	Output 1	Output 2	Output 3
Pattern 1	corr2(1,1)	corr2(1,2)	corr2(1,3)
Pattern 2	corr2(2,1)	corr2(2,2)	corr2(2,3)
Pattern 3	corr2(3,1)	corr2(3,2)	corr2(3,3)

6. Perform the above task (steps 2 to steps 5) using the pseudo inverse rule.

Part 2:

1) 5 face images will be available for download in the lab. Use Hebbian learning rule and pseudo inverse rule to create autoassociative memory.

2) Now add white Gaussian noise with a target SNR of 20 dB to each image. Use the awgn() function given below. Then apply the trained network to recognize the noisy images.

```
def awgn(signal, snr):
    db_signal = 10 * np.log10(np.mean(signal ** 2)) db_noise =
    db_signal - snr
    noise_power = 10 ** (db_noise / 10)
    noise = np.random.normal(0, np.sqrt(noise_power), len(signal)) return signal +
    noise
```

3) Compare the results obtained from pseudo inverse rule and Hebbian learning rule and make a table of correlation coefficients for both rules.

Hint: Load and preprocess all 5 images.

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

b. Convert the RGB color image into grey scale image using PIL.Image.convert('L').

c. Normalize the image into [0, 1] range using sklearn.preprocessing.normalize().

d. Convert the matrix into a column vector using numpy.reshape().

Marking:

Show the following to the TA during the lab period:

Part 1:

- Show the noisy input digits and the network output for the noisy digits.
- Create a table of correlation coefficients between the clean digits and the output from the noisy digits.
- Do this for both the Hebbian and Pseudo-Inverse rules.

Part 2:

- Show the noisy input faces and the network output for the noisy faces.
- Create a table of correlation coefficients between the clean faces and the output from the noisy faces.
- Do this for both the Hebbian and Pseudo-Inverse rules.

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