Lab Assignment #1 – Design and implement Shallow Neural networks to predict/classify

Due Date: Sunday 11:59pm, Week 3

Purpose: The purpose of this Lab assignment is to:

- Design and implement Shallow Neural networks to predict/classify:
  - Practice the use of perceptrons and ADALINE
  - Practice the design and implementation of shallow nets
  - Use perceptron and ADALINE for simple classification problems

References: Read Fausset's text chapter 2, Neural networks and deep learning book, chapter 1, and the lecture slides. This material provides the necessary information that you need to complete the exercises.

Be sure to read the following general instructions carefully:

- This assignment must be completed individually by all the students.
- See the naming and **submission rules** at the end of this document
- You will have to **provide a demonstration video for your solution** and upload the video together with the solution on eCentennial through the assignment link. See the **video recording instructions** at the end of this document.

Exercise 1

Write a Python program to **classify letters from different fonts using delta rule learning**. The training set contains **21 input vectors** (patterns) shown below:

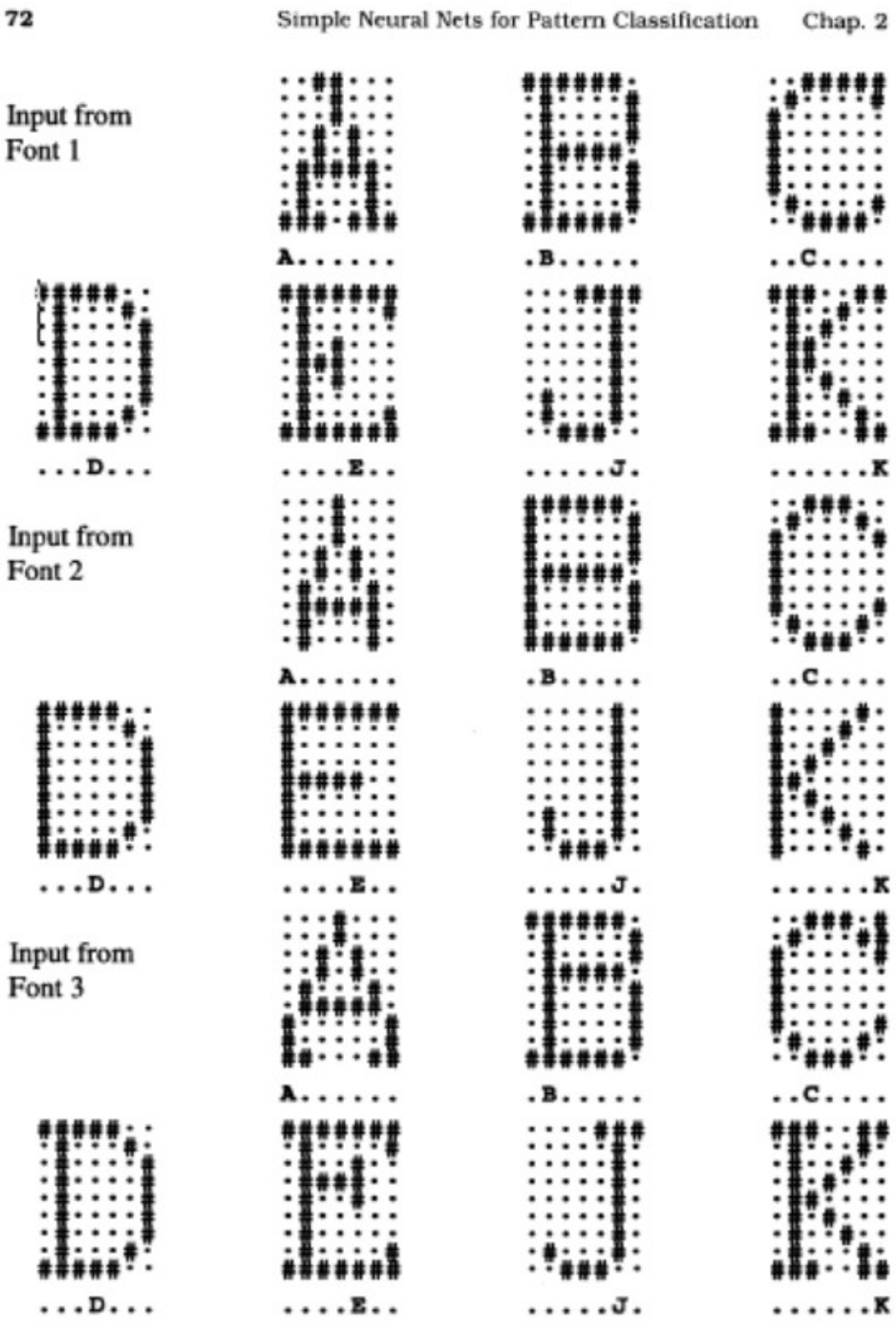


Figure 2.20 Training input and target output patterns.

Each input vector is a **63-tuple** representing a letter expressed as a pattern on a **7 x 9** grid of pixels.

Convert the letters to bipolar form. (You may wish to enter the letters as "2" if the pixel is on and "0" if it is off to facilitate testing with noisy patterns after training; your program should subtract 1 from each component of the input pattern to obtain the bipolar vector.)

For example, the first pattern can be represented by the vector:

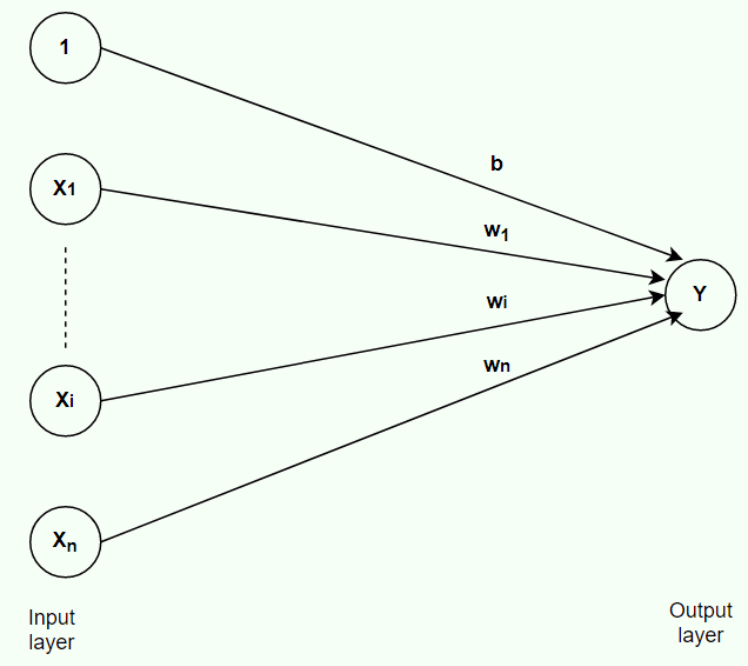
[0,0,2,2,0,0,0, 0,0,0,2,0,0,0, 0,0,0,2,0,0,0,……, 2,2,2,0,2,2,2]

Now you can subtract 1 from each component to obtain the bipolar representation:

[-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]

As the first example of using the perceptron for character recognition, consider the **21 input patterns** in Figure 2.20 as examples of **A** or **not-A (if your first name starts with a letter from A-D inclusively), B or not-B (if your first name starts with a letter from E-K inclusively, C or not-C (if your first name starts with a letter from L-R inclusively), and D or not-D (if your first name starts with a letter from S-Z inclusively).**

In other words, we train the perceptron to classify each of these vectors as belonging, or not belonging, to the class **A** (or B, C, D as stated above). In that case, the target value for each pattern is either **1** or **- 1**; only the first component of the target vector shown is applicable. The net is as shown below, and **n = 63**.



There are three examples of **A** and 18 examples of **not-A** in the training set.

a) First, classify the input using perceptron.
b) Then, classify the input using ADALINE and compare the ability of the trained ADALINE to classify noisy input to the results for the perceptron. Try 5, 10, 15, 20 pixels wrong and the same levels of missing data.

Your output should **display the results of classification in a friendly format and state the differences between two algorithms** regarding handling noise and missing data.

(4 marks)

Exercise 2

Write a Python program to train the model and use the following multilayer ANN to predict the output for class Die_Live:

- Die_Live 1: 1, 0
- Die_Live 2: 0, 1

The set of data is from http://archive.ics.uci.edu/ml/datasets/Hepatitis.
Number of Instances: 155
Number of Attributes: 20 (including the class attribute)

Replace all missing records with a single and unique value, which is the **mean value** of that attribute.

Use a multilayer ANN (19-30-15-2).
Input layer – 19 neurons
Second layer – 30 neurons
Third layer – 15 neurons
Output layer – 2 neurons
Activation function for all layers – sigmoid.

Make the necessary adjustments to the net architecture to achieve a high accuracy.

(6 marks)

Evaluation:

Functionality: - Correct implementation of requirements - Code demonstration and brief explanation in a short video	70% 10%
Algorithm design: - correct design of classes and methods similarly to class examples - Correct use of naming guidelines for classes, variables, methods.	15% 5%
Total	100%

You must **name your Jupyter notebook file** according to the following rule:

YourFullname_COMP258Labnumber_Exercisenumber.

Example: **JohnSmith_ COMP258Lab1_Ex1**

Provide your **student number and full name as a comment** at the top of your code for each exercise.

Submission rules:

Submit your solution as a **zip file** that is named according to the following rule:
YourFullname_ COMP258Labnumber.zip

Example: **JohnSmith_ COMP258Lab1.zip**

Use 7-zip to compress files (https://www.7-zip.org/download.html).

Demonstration Video Recording

Please record a short video (max 3-4 minutes) to demonstrate your assignment solution. You may **use the Windows 10 Game bar** to do the recording:

1. Press the Windows key + G at the same time to open the Game Bar dialog.
2. Check the "Yes, this is a game" checkbox to load the Game Bar.
3. Click on the Start Recording button (or Win + Alt + R) to begin capturing the video.
4. Stop the recording by clicking on the red recording bar that will be on the top right of the program window.
(If it disappears on you, press Win + G again to bring the Game Bar back.)

You'll find your recorded video (MP4 file), under the Videos folder in a subfolder called Captures.

Submit the video together with your solution.