College of Computer Science and Engineering

Department of Computer Science and Artificial Intelligence

CCAI-321: Artificial Neural Networks

Lab#3 Building a Hamming Network using Python

1948080

Marks Obtained = / 10 PLO = S1 - AI

Marks:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Total |
| Allocated | .5 | 1 | 2 | 1.5 | 3 | 2 | 10 |
| Obtained | .5 | 1 | 2 | 1.5 | 3 |  | 8 |
|  |  |  |  |  |  |  |  |
| Allocated |  |  |  |  |  |  |  |
| Marks |  |  |  |  |  |  |  |

Weighted Marks:

|  |  |  |  |
| --- | --- | --- | --- |
| Allocated |  |  |  |
| Obtained |  |  |  |

Manually check if the answers you got for p = [1 -1 -1] is correct. This means, you need to compute by hand the following. Please show ALL steps. Use the same W1, b1, and W2 as in the previous question

Objectives

* Implement a hamming network in python
* Use the Hamming network to classify a toy example in python

Lab Tool(s)

[Download Python | Python.org](https://www.python.org/downloads/)

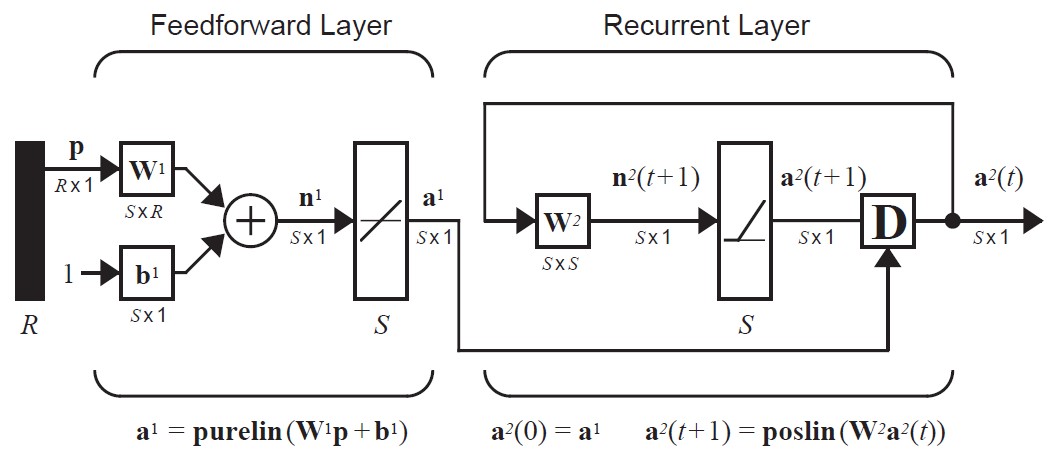
[Anaconda | Individual Edition](https://www.anaconda.com/products/individual)

Lab Deliverables

Submit a pdf document on Blackboard containing your solution to the lab assessment at the end of this document.

What is a Hamming Network?

A Hamming network is used to solve binary pattern recognition problems, where each element in the input vector has one of two possible values. The Hamming network has a feedforward layer and a recurrent layer (as shown in the figure below). The number of neurons in both layers are the same.



In the Hamming network, two transfer functions are used: purelin and poslin. We have implemented these transfer functions, along with others in the previous labs. We have also implemented a multi-input neuron in the previous lab. In this lab we shall use those to develop a simple Hamming Network. The two transfer functions used here are presented below for reference.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Shape | Comments |
| Linear | 𝑝𝑢𝑟𝑒𝑙𝑖𝑛(𝑛) = 𝑛 |  | The output of the function keeps on increasing and never  clips to any value |
| Positive Linear | 0 𝑓𝑜𝑟 𝑛 < 0  𝑝𝑜𝑠𝑙𝑖𝑛(𝑛) = {  𝑛 𝑒𝑙𝑠𝑒 |  | Continuous yet clipped at 0 and linear for +ive values a = max(n,0) |

Q1. Develop a feedforward layer. The feedforward layer is similar to the multiple input neuron you implemented in the last lab. It takes as an input a column vector p , weight matrix w, bias vector b and produces a summation output n which when passed to the transfer function (purelin) generates the output a.

Create a function titled feedforwardlayer and pass the parameters i.e. values of ‘p’, ‘w’, ‘b’ and return the

output ‘a1’ from that function. [0.5 marks]

Q2. Test the feedforwardlayer. In the Hamming network, the weight matrix W contains the prototype patterns. In the orange and apple classification tasks, the prototype patterns are po = [1 -1 -1] and pa = [1 1 -1]. This means we should initialize W1 as

𝟏 = [𝐩𝐓𝐨𝐓] = [𝟏𝟏 −𝟏𝟏 −−𝟏𝟏]

𝐖

𝐩𝐚

The bias in the hamming network is equal to the number of inputs.

𝐛𝟏 = [𝟑]

𝟑

Test the feedforward layer by passing w and b, given the following p values, and write the output a1 in

the table below. [1; 0.25 each]

|  |  |
| --- | --- |
| p | a1 |
| P = [1 -1 -1] |  |
| P = [1 1 -1] |  |
| P = [-1 -1 -1] |  |
| P = [-1 1 -1] |  |

Q3. Develop a recurrent layer. The recurrent layer takes the following inputs: t, a, and W. t is the timestep and a is a1 when t=0; and a2 (t-1) otherwise. To compute the output of this layer, you need to do the following:

First, when t=0, a2 = a1

If t > 0; a2(t+1) poslin(W2a2(t))

Create a function titled recurrentlayer and pass the parameters i.e. values of ‘p’, ‘w’, ‘t’ and return the

output ‘a2’ from that function. [2 marks]

Remember a1 indicates the output of the first later, and a2 is the output of the second layer.

𝟏 −𝟎. 𝟓

Q4. Test the recurrent layer you created by passing W and a1. Let a1 be [4 2], and W be [ ]

−𝟎. 𝟓 𝟏

Remember, in a Hamming network The weight matrix is defined as

𝐖𝟐 = [𝟏 − 𝜺]

−𝜺 𝟏

where 𝜺 = 1 / (S-1) where S is number of neurons.

Fill the table below for t=0, 1 and 2. [1.5; 0.25 each]

|  |  |  |
| --- | --- | --- |
| t | Input | Output |
| 0 |  |  |
| 1 |  |  |
| 2 |  |  |

Q5. Test the complete Hamming network you developed. To do this, first you will pass the input to the feedforward network once, then pass the output of this layer to the recurrent layer. Repeat the last step

until no change in the output occurs. Test your network using the following parameters and input values

(in the table). [3; 0.25 each]

𝟏 −𝟏 −𝟏

W1 = [ ]

𝟏 𝟏 −𝟏

𝟑

b1 = [ ]

𝟑

𝟏 −𝟎. 𝟓

W2 = [ ]

−𝟎. 𝟓 𝟏

|  |  |  |
| --- | --- | --- |
| Input (p) | a2(1) | a2(2) |
| p = [1 1 1] |  |  |
| p = [1 1 -1] |  |  |
| p = [1 -1 1] |  |  |
| p = [1 -1 -1] |  |  |
| p = [-1 1 1] |  |  |
| p = [-1 1 -1] |  |  |

Q6. Manually check if the answers you got for p = [1 -1 -1] is correct. This means, you need to compute by hand the following. Please show ALL steps. Use the same W1, b1, and W2 as in the previous question. [2 points]

a1, the output of the first layer, which is: a1 = purelin(n), where n = W1p+b1 a2(0), the output of the second layer at timestep 0 a2(1), the output of the second layer at timestep 1

a2(2), the output of the second layer at time step 2, where a2(0) = a1, and a2(t+1) = poslon(n), n = W2p