**Homework-4**

**Name: Chitradevi Maruthavanan**

**PSU\_ID:** 950828319

1. Consider the neural network (1) with a scalar input x and parameters.

WH = , bH  = WO = [-1,2] , bO = 0.5

using a hard threshold activation function (2) and threshold output function (4).

(a)What is Nh, the number of hidden units? What is No, the number of output units?

**WH = Nh \* Ni, Since the parameter matrix WH is a 2\*1 matrix, so the number of hidden units Nh is 2. Wo = No \* Nh, Since the parameter matrix W0 is a 1\*2 matrix, so the number of output units N0 is 1.**

(b) Write zH in terms of x. Show the functions for each component ZjH

**zH = wH x +bH**

**zH = x +**

**zH1= x-1**

**zH2= x-3**

(c) Write uH in terms of x. Show the functions for each component **uHj.**

**uH = gact(zH)**

**uH1 = gact(zH1) = gact(x-1)**

**uH2= gact(ZH2) = gact(x-3)**

**uH1 = 0 , if x -1 < 0; uH1 = 1 , if x -1 >=0;**

**uH2 = 0 , if x -3 < 0; uH2 = 1 , if x -3 >=0;**

**To simplify,**

**uH1 = 0 , if x < 1; uH1 = 1 , if x >=1;**

**uH2 = 0 , if x < 3; uH2= 1 , if x >=3;**

(d) Write zO in terms of x.

**zO = wouH+bo**

**zO = [-1 2] [uH1, uH2] + 0.5**

**zO = [-1 2] [x < 1, 1 <= x < 3, x >= 3] + 0.5**

**substituting uH1 and uH2 in terms of x, we get:**

**zO = [-1 2] [0 0] + 0.5 if x < 1**

**zO = [-1 2] [1 0] + 0.5 if 1 <= x < 3**

**zO = [-1 2] [1 1] + 0.5 if x >= 3**

**To Simplify we get,**

**zO = 0.5 if x < 1**

**zO = - 0.5 if 1 <= x < 3**

**zO = 1.5 if x >= 3**

(e) What is y^ in terms of x?

**gout(zo)**

**gout(zo) = 1 if zo >= 0**

**gout(zo) = 0 if zo < 0**

**Substituting zO, we get:**

**1, if [x < 1, x >= 3]**

**0, if [1 <= x < 3]**

1. Consider the data set for four points with scalar features xi and binary class labels

yi = 0; 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | xi | 0 | 1 | 3 | 5 |  |
|  |  | yi | 0 | 0 | 1 | 0 |  |

(a) Find a neural network with Nh = 2 units, No = 1 output units such that ^yi = yi  on all four data points. Use a network similar in structure to the previous problem. Also, you want to find features that can distinguish between x = 3 and x = {0, 1;,5}. Since there are many features, use two features: whether x >= 2 and x >= 4. Use a hard threshold activation function (2) and threshold output function (4). State the weights and biases used in each layer.

**To determine the weights and biases for this network,**

**For the hidden layer, we can set:**

* **WH 1 = WH 2 = 1**
* **bH 1 = -2 (so that gact(x - 2) = 1 when x >= 2, and gact(x - 2) = 0 otherwise)**
* **bH 2 = -4 (so that gact(x - 4) = 1 when x >= 4, and gact(x - 4) = 0 otherwise)**

**For the output layer, we can set:**

* **wO = [1, -1]**
* **bO = -0.5**

**Putting it all together, the neural network has the following weights and biases:**

**WH = bH = WO = [1, -1] bO = -0.5**

**The above weights and biases arrived on a trial and error basis by trying different xi and making sure the correct ^yi was achieved. The detailed computation is described below:**

(b) Compute the values of ^yi and all the intermediate variables zHi, uHi and zOi for each sample x = xi.

**We can compute the intermediate variables for each sample:**

* **X = 0**

**zH = wH x +bH**

**zH = x +**

**zH1= +**

**zH11 = -2, zH12 = -4**

**uH = gact(zH)**

**uH11 = gact(zH11) = gact(-2)**

**uH12= gact(ZH12) = gact(-4)**

**uH11 = 0 uH12= 0**

**zO = wouH+bo**

**zO1= [1 -1] [uH11, uH12] - 0.5**

**zO1 = [1 -1] [0, 0] - 0.5**

**zO1 = 0+0-0.5**

**zO1= -0.5**

**gout(zo)**

**gout(zo) = 1 if zo >= 0**

**gout(zo) = 0 if zo < 0**

**gout(-0.5) = 0**

* **X = 1**

**zH2 = +**

**zH21 = -1, zH22 = -3**

**uH21 = gact(-1), uH22 = gact(-3)**

**uH21 = 0, uH22= 0**

**zO2= [1 -1] [0, 0] - 0.5**

**zO2= 0+0-0.5**

**zO2= -0.5**

**gout(-0.5)**

**0**

* **X = 3**

**zH3 = +**

**zH31 = 1, zH32 = -1**

**uH31 = gact(1), uH32 = gact(-1)**

**uH31 = 1, uH32= 0**

**zO3= [1 -1] [1, 0] - 0.5**

**zO3 = 1+0-0.5**

**zO3 = 0.5**

**gout(0.5)**

**1**

* **X = 5**

**zH4 = +**

**zH41 = 3, zH42 = 1**

**uH41 = gact(3), uH42 = gact(1)**

**uH41 = 1, uH42= 1**

**zO4 = [1 -1] [1, 1] - 0.5**

**zO4 = 1-1-0.5**

**zO4 = -0.5**

**gout(-0.5)**

**0**

(c) Now suppose we are given a new sample, x = 3:5. What does the network predict as ^y?

* **X = 3.5**

**zH = +**

**zH1 = 1.5, zH2 = -0.5**

**uH1 = gact(1.5), uH2 = gact(-0.5)**

**uH1 = 1, uH2= 0**

**zO = [1 -1] [1, 0] - 0.5**

**zO = 1-0-0.5**

**zO = 0.5**

**gout(zo)**

**gout(0.5)**

**1**

3(a) Write the components of zH and uH as a function of (x1, x2). For each component j,

indicate where in the (x1; x2) plane uHj= 1.

**zH = wH x +bH**

**zH  = [x1 x2]+**

**zH1 = x1**

**zH2 = x2**

**zH3 = x1+x2-1**

**uH = 1 if zH >= 0**

**uH = 0 if zH < 0**

**uH1= 1 if x1 >= 0; uH1= 0 if x1 < 0**

**uH2= 1 if x2 >= 0; uH2= 0 if x2 < 0**

**uH3= 1 if x1+x2 >= 1; uH3= 0 if x1+x2 < 1**

(b) Write zO as a function of (x1; x2). In what region is yˆ= 1?

**zO = wouH+bo**

**zO = [1 1 -1] [uH1 uH2  uH3] – 1.5**

**gout(zo)**

**gout(zo) = 1 if zo >= 0**

**zO =[1 1 -1] [ uH1 , uH2 ,uH3] – 1.5**

**zO = [1 1 -1] [1 1 0] – 1.5 if x1 >= 0, x2 >= 0 and x1+x2 < 1**

**zO = [1+1-0]-1.5. = 0.5**

**gout(0.5) = 1**

**1 only when x1 >= 0 , x2 >= 0, x1+x2 < 1**

**The other cases lead to yˆ= 0. They are:**

**zO = [0+0-0]-1.5. = -1.5 if x1 < 0, x2 < 0 and x1+x2 < 1**

**zO = [1+0-0]-1.5. = -0.5 if x1 >=0, x2 < 0 and x1+x2 < 1**

**zO = [0+1-0]-1.5. = -0.5 if x1 < 0, x2 >= 0 and x1+x2 < 1**

**zO = [0+0-1]-1.5. = -2.5 if x1 < 0, x2 < 0 and x1+x2 >= 1**

**zO = [0+1-1]-1.5. = -1.5 if x1 < 0, x2 >= 0 and x1+x2 >= 1**

**zO = [1+0-1]-1.5. = -1.5 if x1 >=0, x2 < 0 and x1+x2 >=1**

**zO = [1+1-1]-1.5. = -0.5 if x1 >=0, x2 >= 0 and x1+x2 >= 1**

**1 only when x1 >= 0 , x2 >= 0, x1+x2 < 1**

4. Architecture choices for different problems: For each problem, state possible selections for the

dimensions Ni, Nh, No and the functions gact(.) and gout(.). Indicate which parameters are

free to choose.

(a) One wants a neural network to take as an input a 20\*20 gray scale image and determine

which letter (`a' to `z') the image is of.

**Possible choices for architecture are:**

* **Ni = 400 (20 x 20 pixels)**
* **Nh = variable, free to choose**
* **No = 26 (one output per letter)**
* **gact(·) = ReLU or sigmoid (free to choose)**
* **gout(·) = softmax function for k- class classification, as we want a probability distribution over the 26 classes**

(b) One extracts 120 features of a sample of a speech recording (like the MFCCs). Based on

the audio samples, the network is to determine if the speech is male or female.

**Possible choices for architecture are:**

* **Ni = 120 (number of features)**
* **Nh = variable, free to choose**
* **No = 1 (binary classification)**
* **gact(·) = ReLU or sigmoid (free to choose)**
* **gout(·) = sigmoid for binary classification, as we want to output a probability between 0 and 1 indicating the likelihood of the sample being from a male or female speaker**

(c) One wants a neural network to predict the stock price based on the average stock price

of the last five days.

**Possible choices for architecture are:**

* **Ni = 5 (last 5 days)**
* **Nh = variable, free to choose**
* **No = 1 (stock price prediction)**
* **gact(·)= ReLU or sigmoid (free to choose)**
* **gout(·) = linear, as we want to output a real-valued prediction for the stock price**

5. Implementation in python: Write python code for implementing the following steps for a

batch of samples:

1. The hidden layer step (10a).

**import numpy as np**

**def hidden\_layer(X, WH, BH, gact):**

**ZH = np.dot(X, WH.T) + BH**

**UH = gact(ZH)**

**return UH, ZH**

(b) The output layer step (10b) for binary classification with a sigmoid output (3).

**import numpy as np**

**def sigmoid(x):**

**return 1 / (1 + np.exp(-x))**

**def binary\_output\_layer(UH, WO, BO):**

**ZO = np.dot(UH, WO.T) + BO**

**UO = sigmoid(ZO)**

**return UO**

(c) The output layer step (10b) for K-class classification with a softmax output (5).

**import numpy as np**

**def softmax(x):**

**exp\_x = np.exp(x)**

**return exp\_x / np.sum(exp\_x, axis=1, keepdims=True)**

**def output\_layer\_softmax(UH, WO, BO):**

**ZO = np.dot(UH, WO.T) + BO**

**UO = softmax(ZO)**

**return UO**

For all examples, avoid for-loops and instead use Python broadcasting.