**CSE 5693 Machine Learning**

**HW3 Artificial Neural Network Learning**

Josias Moukpe

Written Assignment

1. **4.1. What are the values of weights w0, w1, and w2 for the perceptron whose decision surface is illustrated in Figure 4.3? Assume the surface crosses the x1 axis at -1, and the x2 axis at 2.**

Answer:

Decision boundary is E: w0 + w1x1 + w2x2 > 0 and pass by points P1 (-1, 0) and P2 (0, 2). We also have P3 (-2, 3) as a positive instance and P4 (0, 0) as a negative instance.

Substituting P1 and P2 in E, we get:

w0 + -w1 =0 => **w1 = w0**

w0 + 2\*w2 = 0 => **w2 = -1/2 \* w0**

So, we have E is also w0 + w0x1 – 1/2\*w0\*x2 = 0.

Moreover, since P3 is a positive example and P4 is a negative example, for the perceptron decision boundary, we find **w0 ≤ -1**.The solution is then:

**For all w0 ≤ -1, w1 = w0, w2 = -1/2 \* w0.**

An instance would be w0 = -1, w1 = -1, w2 = 1/2

1. **4.2. Design a two-input perceptron that implements the boolean function A ^ - B. Design a two-layer network of perceptrons that implements A XOR B.**

Answer:

|  |  |  |
| --- | --- | --- |
| A | B | H1 = A ^ - B |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Let our perceptron be of the form w0 + w1A + w2B > 0 with a step activation function which return 1 when w0 + w1A + w2B > 0 and 0 otherwise. The perceptron takes for input the values A and B. To solve for our perceptron, let’s find w0, w1, w2.

For A = 0, B = 0, the result should be 0 so w0 + 0 + 0 ≤ 0, so let’s take **w0 = -1**

For A = 0, B = 1, the result should be 0 so -1 + w2 ≤ 0, so let’s take **w2 = -1**

For A = 1, B = 0, the result should be 1 so -1 + w1 > 0, so let’s take **w1 = 2**

For A = 1, B = 0, the result should be 0 so -1 + 2 -1 ≤0, which it already is so our choice of w0, w1, w2 are valid for implementing A ^ -B.

|  |  |  |
| --- | --- | --- |
| A | B | A XOR B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

A XOR B = (A ^ -B) v (-A ^ B) so we have the combination of three 2 inputs perceptrons. 2 on the first layer, and one on the second layer.

1st Layer:

We already found from previous part of the question that (A ^ -B) is implemented by a perceptron with w00 = -1, w01 = 2, w02 = -1.

|  |  |  |
| --- | --- | --- |
| A | B | H2 = -A ^ B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

For part (-A ^ B), we can flip the value of w1 and w2 and we get w10 = -1, w11 = -1, w12 = 2

2nd Layer is a disjunction so we have

|  |  |  |
| --- | --- | --- |
| H1 | H2 | H1 v H2 |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

For the disjunction, we can use the table and get w20 = -1, w21 = 2, w22 = 2 for the output layer

1. **4.9. Recall the 8 x 3 x 8 network described in Figure 4.7. Consider trying to train a 8 x 1 x 8 network for the same task; that is, a network with just one hidden unit. Notice the eight training examples in Figure 4.7 could be represented by eight distinct values for the single hidden unit (e.g., 0.1,0.2, . . . ,0.8). Could a network with just one hidden unit therefore learn the identity function defined over these training examples? Hint: Consider questions such as "do there exist values for the hidden unit weights that can create the hidden unit encoding suggested above?” "do there exist values for the output unit weights that could correctly decode this encoding of the input?” and "is gradient descent likely to find such weights?”**

****

Answer: Run the program with 1 unit hidden layer to find out.

1. **With the programming assignment:**
   1. **discuss the hidden values in testIdentity using 3 and 4 hidden units (Why do 4 hidden units also work? What do the hidden values represent? Any significant difference in the number of iterations to convergence and why?)**

Answer:

1. **compare performance of using validation set to not using it in testIrisNoisy. Include a plot for the comparisons.**

Answer: