Assignment 2 Machine Learning COS4852

Adriaan Louw (53031377)

July 13, 2018

Contents

1 Question 1

1.1 Question 1(a)

Firstly we calculate the line

$$x_2 = mx_1 + c \tag{1}$$

for the intersect points (2,0) and (0,6).
Calculating slope m,

$$\begin{aligned} m &= \frac{6-0}{0-2} \\ &= -3 \end{aligned} \tag{2}$$

x_2 intercept c is 6.
This makes equation 1

$$x_2 = -3x_1 + 6 \tag{3}$$

Nils J Nilsson (1998) gives the equation for the hyperplane as

$$\sum_{i=1}^{n} x_i \omega_i \geq \theta \tag{4}$$

which in this case gives the equation for the hyperplane to be

$$\omega_1 x_1 + \omega_2 x_2 + \omega_3 = 0 \tag{5}$$

We need to get equation 5 in the form of equation 1

$$\begin{aligned} \omega_1 x_1 + \omega_2 x_2 + +\omega_3 &= 0 \\ \omega_2 x_2 &= -\omega_1 x_1 - \omega_3 \\ x_2 &= -\frac{\omega_1 x_1}{\omega_2} - \frac{\omega_3}{\omega_2} \end{aligned} \tag{6}$$

Comparing coefficients m and c from equation 3 to 6 we get

$$\begin{aligned} -\frac{\omega_1}{\omega_2} &= -3 \\ \omega_1 &= 3\omega_2 \end{aligned} \tag{7}$$

and

$$\begin{aligned} -\frac{\omega_3}{\omega_2} &= 6 \\ \omega_3 &= -6\omega_2 \end{aligned} \tag{8}$$

If we choose $\omega_3 = -2$ then $\omega_1 = 1$ and $\omega_2 = \frac{1}{3}$. This makes the hyperplane equation from equation 5

$$x_1 + \frac{x_2}{3} - 2 = 0 \tag{9}$$

Now we need to test this hyperplane. For positive instance (2,6)

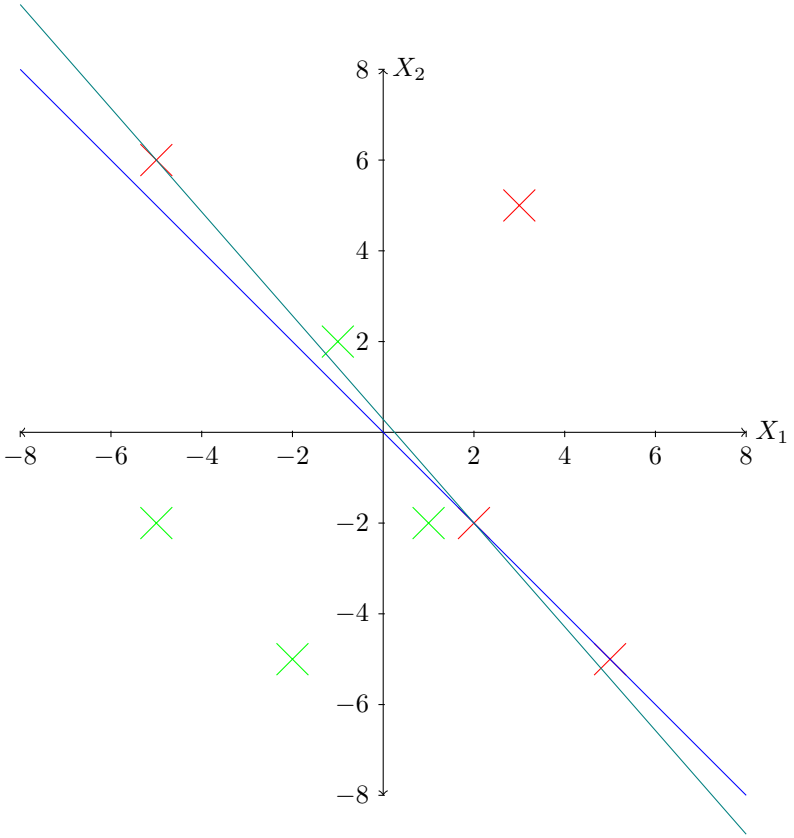
$$\begin{aligned} x_1 + \frac{x_2}{3} - 2 &= \\ 2 + \frac{6}{2} - 2 &= \\ 2 & \end{aligned} \tag{10}$$

Which is as expected.
And the negative instance (-1,2)

$$\begin{aligned} x_1 + \frac{x_2}{3} - 2 &= \\ -1 + \frac{2}{3} - 2 &= \\ -\frac{7}{3} & \end{aligned} \tag{11}$$

This is also as expected. The perceptron now classifies the the data correctly

1.2 Question 1(b)



From the above image we can see that any that it is not possible to create a hyperplane that correctly classifies all negative instances and positive instances. The blue line is the line $x_2 = -x_1$ and the teal line is the line $x_2 = -\frac{7}{8}x_1 + \frac{2}{7}$. The any minimum plane that correctly classifies all the negative instances will classify the positive instance (-1,2) incorrectly as negative.

We can create a hyperplane from regression from all the points close to where the hyperplane should be. Using negative points (-5,6),(2,-2),(5,-5) and positive points (-1,2),(1,-2)

For the equation of the regressed line $x_2 = mx_1 + c$

$$
\begin{aligned}
m &= r\frac{S(x_2)}{S(x_1)} \\
m &= \frac{\sum((x_1 - \bar{x_1})(x_2 - \bar{x_2}))}{\sqrt{\sum(x_1 - \bar{x_1})^2 \sum(x_2 - \bar{x_2})^2}} \frac{\sqrt{\frac{\sum(x_2 - \bar{x_2})^2}{n-1}}}{\sqrt{\frac{\sum(x_1 - \bar{x_1})^2}{n-1}}}
\end{aligned} \tag{12}
$$

where r is Pearsons Correlation Coefficient and S is standard deviation of axis x_2 or x_1.

Here follows the calculation

x_1	x_2	$x_1 - \bar{x_1}$	$x_2 - \bar{x_2}$	$(x_1 - \bar{x_1})(x_2 - \bar{x_2})$	$(x_1 - \bar{x_1})^2$	$(x_2 - \bar{x_2})^2$
-5	6	-5.4	6.2	-33.48	29.16	38.44
2	-2	1.6	-1.8	-2.88	2.56	3.24
5	-5	4.6	-4.8	-22.08	21.16	23.04
-1	2	-1.4	2.2	-3.08	1.96	4.84
1	-2	0.6	-1.8	-1.08	0.36	3.24

From the above table we have $\bar{x_1} = 0.4, \bar{x_2} = -0.2, \sum((x_1 - \bar{x_1})(x_2 - \bar{x_2})) = -62.6, (x_1 - \bar{x_1})^2 = 55.2, and (x_2 - \bar{x_2})^2 = 72.8$

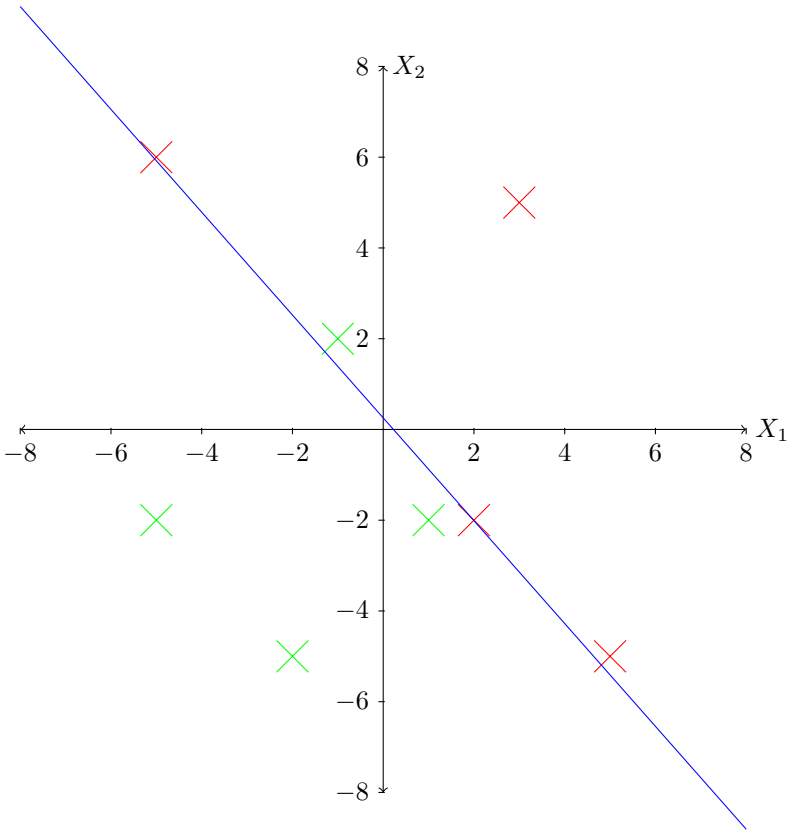
Passing these into equation 12 we get m = -1.13

$$\begin{aligned} c &= \bar{x_2} - m\bar{x_1} \\ &= -0.2 - (-1.13)(0.4) \\ &= 0.25 \end{aligned} \tag{13}$$

Which gives us equation

$$x_2 = -1.13x_1 + 0.25 \tag{14}$$

Visually it would be



This will minimise the error even though it would incorrectly classify (-1,2)

1.3 Question 1(c)

1.4 Question 1(d)

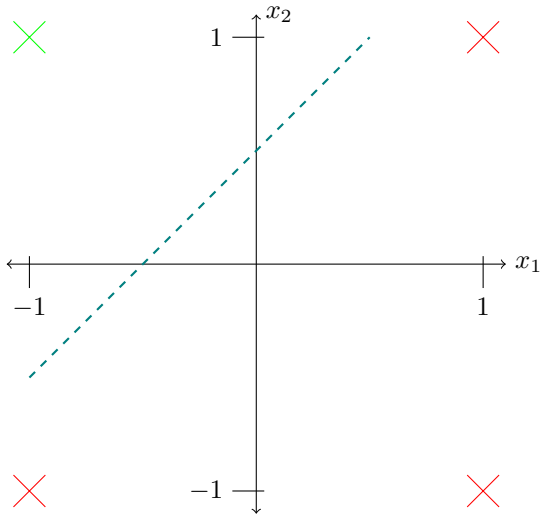
2 Question 2

2.1 Question 2(a)

The truth table for this function is

x_1	x_2	$\neg x_1$	$\neg x_1 \vee x_2$
-1	-1	1	-1
-1	1	1	1
1	-1	-1	-1
1	1	-1	-1

Visually this is:



The dashed line represents the function

$$x_2 = x_1 + 0.5 \tag{15}$$

This hyperplane will classify the boolean function correctly because it linearly separates all the positive instances from the negative ones.

The equation for the hyperplane with weights are given by equation 5 and we know from equation 6 what the relationship is from the weight equation to the line equation. Thus for case

$$\begin{aligned} m &= -\frac{\omega_1}{\omega_2} \\ 1 &= -\frac{\omega_1}{\omega_2} \\ \omega_2 &= -\omega_1 \end{aligned} \tag{16}$$

and

$$\begin{aligned} c &= -\frac{\omega_0}{\omega_2} \\ 0.5 &= -\frac{\omega_0}{\omega_2} \\ \omega_0 &= -\frac{\omega_2}{2} \end{aligned} \tag{17}$$

Now we have the relationship between the weights. If we try $\omega_0 = 1$ we get $\omega_1 = 2$ and $\omega_2 = -2$

Which makes our weight equation

$$2x_1 - 2x_2 + 1 \tag{18}$$

Testing for polarity with point (-1,-1) we get

$$2(-1) - 2(-1) + 1 = 1 \tag{19}$$

which indicates a positive result but we expect a negative result. Thus our polarity is wrong

We try $\omega_0 = -1$ which gives $\omega_1 = -2$ and $\omega_2 = 2$

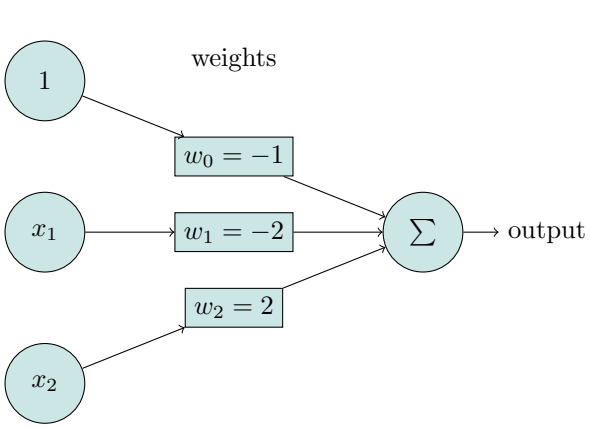
Which makes the weight equation

$$-2x_1 + 2x_2 - 1 \tag{20}$$

Trying all our data we can sum it up in the following table

x_1	x_2	Result
-1	-1	-1
-1	1	4
1	-1	-5
1	1	-1

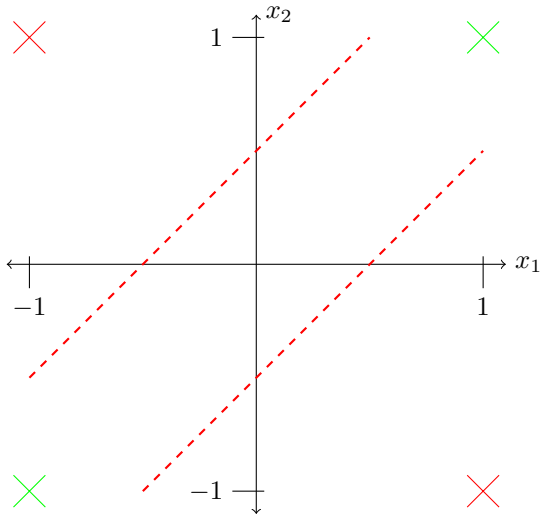
This shows that weights $\omega_0 = -1$; $\omega_1 = -2$ and $\omega_2 = 2$ correctly classify the data



2.2 Question 2(b)

x_1	x_2	$x_1 \bigoplus x_2$	$f_2 = \neg(x_1 \bigoplus x_2)$
-1	-1	-1	1
-1	1	1	-1
1	-1	1	-1
1	1	-1	1

Visually this is:



We can see from the previous diagram that f_2 cannot be linearly separated by 1 hyperplane. In other words a single perceptron cannot classify this function. We need to try to decompose f_2 into multiple linearly separable functions that can be modelled by multiple perceptrons.

$$\begin{aligned} f_2 &= \neg(x_1 \bigoplus x_2) \\ &= \neg((x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)) \\ &= \neg(x_1 \wedge \neg x_2) \wedge \neg(\neg x_1 \wedge x_2) \\ &= (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \end{aligned} \tag{21}$$

We can thus create new function which is equivalent to f_2

$$g(h_1, h_2) = h_1 \wedge h_2 \tag{22}$$

where

$$h_1 = \neg x_1 \vee x_2 \tag{23}$$

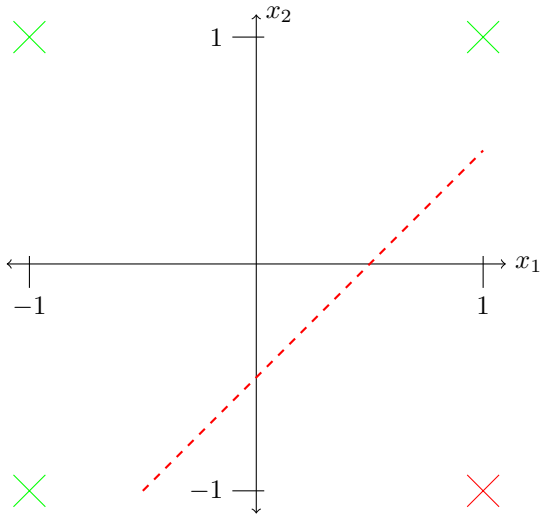
and

$$h_2 = x_1 \vee \neg x_2 \tag{24}$$

The truth table will thus become

x_1	x_2	$x_1 \bigoplus x_2$	$f_2 = \neg(x_1 \bigoplus x_2)$	h_1	h_2	$g(h_1, h_2)$
-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

Drawing h_1 gives



Using equation 5 and knowing this hyperplane goes through points (-0.5,-1) and (1.0,0.5). We have

$$\begin{aligned} \omega_1(-0.5) + \omega_2(-1) + \omega_0 &= 0 \\ \frac{-\omega_1}{2} - \omega_2 + \omega_0 &= 0 \end{aligned} \tag{25}$$

for the point (-0.5,-1) and

$$\begin{aligned} \omega_1(1) + \omega_2(0.5) + \omega_0 &= 0 \\ \omega_1 + \frac{\omega_2}{2} + \omega_0 &= 0 \end{aligned} \tag{26}$$

for point (1.0,0.5)
Equating equations 25 and 26 we get

$$\begin{aligned} \frac{-\omega_1}{2} - \omega_2 + \omega_0 &= \omega_1 + \frac{\omega_2}{2} + \omega_0 \\ \frac{3\omega_1}{2} &= \frac{3\omega_2}{2} \\ \omega_2 &= -\omega_1 \end{aligned} \tag{27}$$

Passing equation 27 into 26 we get

$$\begin{aligned} \omega_1 + \frac{-\omega_1}{2} + \omega_0 &= 0 \\ \omega_0 &= -\frac{\omega_1}{2} \end{aligned} \tag{28}$$

If we take $\omega_0 = 1$ then $\omega_1 = -2$ and $\omega_2 = 2$.

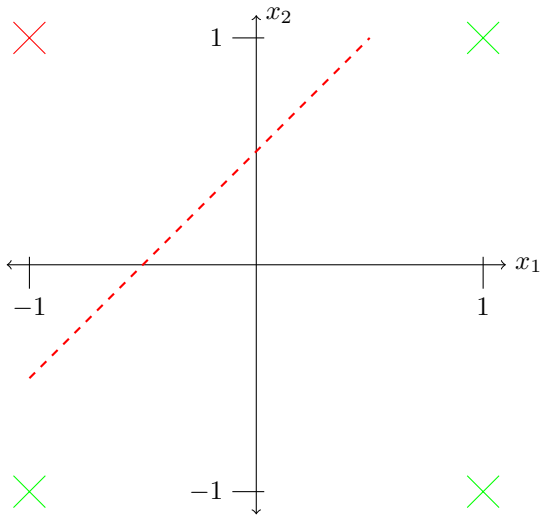
Testing for polarity we use the point (-1,-1) where we expect a positive answer

For point (-1,-1)

$$\begin{aligned} -2x_1 + 2x_2 + 1 = \\ -1(-1) + 2(-1) + 1 = 1 \end{aligned} \tag{29}$$

Which is greater than 0 as expected. Thus the weights for h_1 is $\omega_0 = 1; \omega_1 = -2; \omega_2 = 2$

Now drawing h_2



This is a similar perceptron from as in Question 2a. It uses the same hyperplane but the polarity is reversed. Thus from question 2a $\omega_2 = -\omega_1; \omega_0 = -\frac{\omega_2}{2}$.

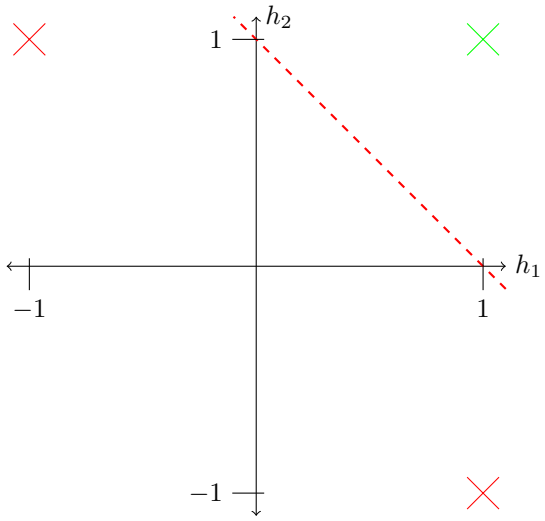
Testing polarity with $\omega_0 = 1$ then $\omega_1 = 2; \omega_2 = -2$. This gives with positive instance (-1,-1)

$$\begin{aligned} 2x_1 - 2x_2 + 1 = \\ 2(-1) - 2(-1) + 1 = 1 \end{aligned} \tag{30}$$

Which indicates the preceptron calssifies the point (-1,-1) correctly as a positive instance

This means $\omega_0 = 1; \omega_1 = 2; \omega_2 = -2$ for h_2

Now we need to create a perceptron for function g. The domain of g as a function of h_1 and h_2 can be shown as follows



From the above diagram we can see that a hyperplane through the points (0,1) and (1,0) will classify this perceptron

From applying equation 5 to point $(h_1, h_2) = (1, 0)$

$$\begin{aligned} \omega_1 x_1 + \omega_2 x_2 + \omega_0 &= 0 \\ \omega_1(1) + \omega_2(0) + \omega_0 &= 0 \\ \omega_1 &= -\omega_0 \end{aligned} \tag{31}$$

and for point $(h_1, h_2) = (0, 1)$

$$\begin{aligned} \omega_1 x_1 + \omega_2 x_2 + \omega_0 &= 0 \\ \omega_1(0) + \omega_2(1) + \omega_0 &= 0 \\ \omega_2 &= -\omega_0 \end{aligned} \tag{32}$$

Now equation equations 31 and 32 we get $\omega_1 = \omega_2$

If we let $\omega_0 = -1$ then from substitution into equations 31 and 32 we get $\omega_1 = \omega_2 = 1$.

Testing for polarity we have positive instance (1,1)

$$\begin{aligned} \omega_1 x_1 + \omega_2 x_2 + \omega_0 &= \\ x_1 + x_2 - 1 &= \\ (1) + (1) - 1 &= 1 \end{aligned} \tag{33}$$

Which is positive as expected

Figure 1 shows the whole network where ω_{1i} denotes the weights for the perceptron modeling h_1, ω_{2i} denotes the weights for the perceptron of h_2 and ω_{3i} denotes the weights of the final perceptron.

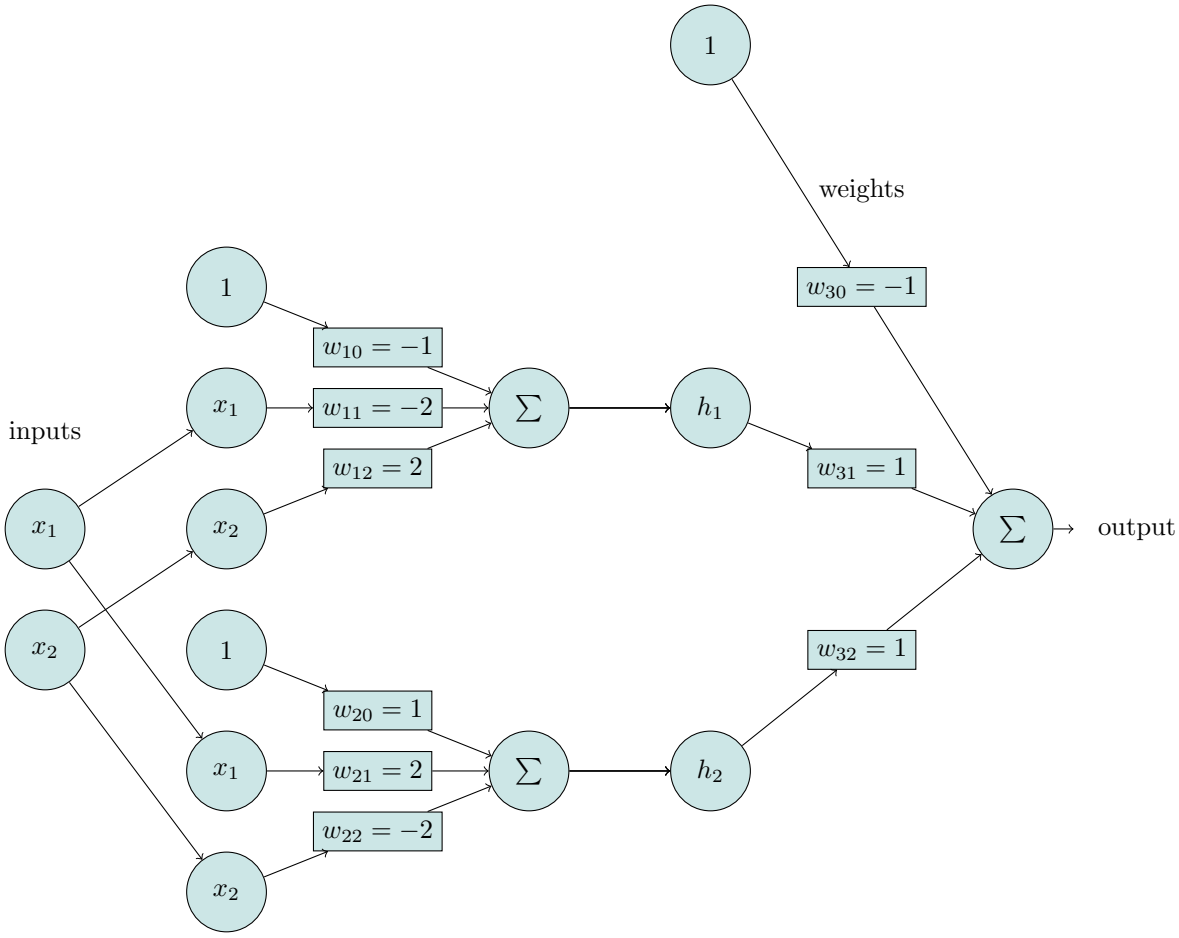


Figure 1: Network for question 2b

ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	b_1	b_2
0.15	0.2	0.25	0.3	0.4	0.45	0.5	0.55	0.35	0.6

Table 1: Start weights

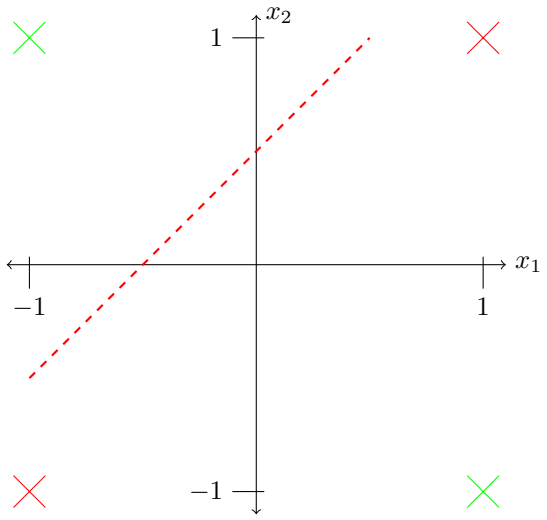
pattern	x_1	x_2	d_1	d_2
p_1	0.1	0.1	0.1	0.9
p_2	0.1	0.9	0.9	0.1
p_3	0.9	0.1	0.9	0.1
p_4	0.9	0.9	0.1	0.9

Table 2: Input Patterns

2.3 Question 2(c)

x_1	x_2	$\neg x_1$	$\neg x_2$	$x_1 \vee x_2$	$\neg x_1 \vee \neg x_2$	$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$
-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

Visually this is:



3 Question 3

Rumelhart, Hinton, and Williams (1986) is the original article introducing the Backpropagation algorithm. The URL for this document is in the reference.

3.1 Question 3(a)

We start with the following weights (Table 1).

3.1.1 Pattern 1

Doing the forward pass. Starting with the values on the hidden units

$$\begin{aligned} net_{h1} &= \omega_1 x_1 + \omega_2 x_2 + b_1 \\ &= 0.15(0.1) + 0.2(0.1) + 0.35 \\ &= 0.385 \end{aligned} \tag{34}$$

$$\begin{aligned} out_{h1} &= \frac{1}{1 + e^{-net_{h1}}} \\ &= \frac{1}{1 + e^{-0.385}} \\ &= 0.595 \end{aligned} \tag{35}$$

$$\begin{aligned} net_{h2} &= \omega_3 x_1 + \omega_4 x_2 + b_1 \\ &= 0.25(0.1) + 0.3(0.1) + 0.35 \\ &= 0.405 \end{aligned} \tag{36}$$

$$\begin{aligned} out_{h2} &= \frac{1}{1 + e^{-net_{h2}}} \\ &= \frac{1}{1 + e^{-0.405}} \\ &= 0.6 \end{aligned} \tag{37}$$

Continuing with the outputs of the network

$$\begin{aligned} net_{o1} &= \omega_5 out_{h1} + \omega_6 out_{h2} + b_2 \\ &= 0.4(0.595) + 0.45(0.6) + 0.6 \\ &= 1.108 \end{aligned} \tag{38}$$

$$\begin{aligned} out_{o1} &= \frac{1}{1 + e^{-net_{o1}}} \\ &= \frac{1}{1 + e^{-1.108}} \\ &= 0.752 \end{aligned} \tag{39}$$

$$\begin{aligned} net_{02} &= \omega_7 out_{h1} + \omega_8 out_{h2} + b_2 \\ &= 0.5(0.595) + 0.55(0.6) + 0.6 \\ &= 1.227 \end{aligned} \tag{40}$$

$$\begin{aligned} out_{o1} &= \frac{1}{1 + e^{-net_{h1}}} \\ &= \frac{1}{1 + e^{-1.223}} \\ &= 0.773 \end{aligned} \tag{41}$$

Now we need to calculate the error

$$\begin{aligned} E_{total} &= E_{o1} + E_{02} \\ &= \frac{1}{2}(0.1 - 0.752)^2 + \frac{1}{2}(0.9 - 0.773)^2 \\ &= 0.220 \end{aligned} \tag{42}$$

$$\begin{aligned} \delta_{o1} &= -(target_{o1} - out_{o1}out_{o1}(1 - out_{o1}) \\ &= -(0.1 - 0.752)(0.752)(1 - 0.752) \\ &= 0.122 \end{aligned} \tag{43}$$

$$\begin{aligned} \frac{\partial E_{total}}{\partial \omega_5} &= \delta_{o1} out_{h1} \\ &= 0.122(0.595) \\ &= 0.073 \end{aligned} \tag{44}$$

We calculate the new value for ω_5

$$\begin{aligned} \omega_5^1 &= \omega_5^0 - \eta \frac{\partial E_{total}}{\partial \omega_5} \\ &= 0.4 - 0.5(0.073) \\ &= 0.364 \end{aligned} \tag{45}$$

$$\begin{aligned} \frac{\partial E_{total}}{\partial \omega_6} &= \delta_{o1} out_{h2} \\ &= 0.122(0.6) \\ &= 0.073 \end{aligned} \tag{46}$$

We calculate the new value for ω_6

$$\begin{aligned} \omega_6^1 &= \omega_6^0 - \eta \frac{\partial E_{total}}{\partial \omega_6} \\ &= 0.45 - 0.5(0.073) \\ &= 0.414 \end{aligned} \tag{47}$$

$$\begin{aligned} \delta_{o2} &= -(target_o 2 - out_{o2}out_{o2}(1 - out_{o2}) \\ &= -(0.9 - 0.773)(0.773)(1 - 0.773) \\ &= -0.022 \end{aligned} \tag{48}$$

$$\begin{aligned} \frac{\partial E_{total}}{\partial \omega_7} &= \delta_{o2} out_{h1} \\ &= -0.022(0.595) \\ &= -0.013 \end{aligned} \tag{49}$$

We calculate the new value for ω_7

$$\begin{aligned}\omega_7^1 &= \omega_7^0 - \eta\frac{\partial E_{total}}{\partial \omega_7} \\ &= 0.5 - 0.5(-0.013) \\ &= 0.507\end{aligned} \tag{50}$$

$$\begin{aligned}\frac{\partial E_{total}}{\partial \omega_8} &= \delta_{o2} out_{h2} \\ &= -0.022(0.6) \\ &= -0.013\end{aligned} \tag{51}$$

We calculate the new value for ω_6

$$\begin{aligned}\omega_6^1 &= \omega_6^0 - \eta\frac{\partial E_{total}}{\partial \omega_6} \\ &= 0.55 - 0.5(-0.013) \\ &= 0.557\end{aligned} \tag{52}$$

Now we need to pass the error back to the hidden units

$$\begin{aligned}\delta_{h1} &= (\delta_{o1}\omega_5^0 + \delta_{o2}\omega_7^0)(out_{h1})(1 - out_{h1}) \\ &= (0.122(0.4) - 0.022(0.5))(0.595)(0.595) \\ &= 0.009\end{aligned} \tag{53}$$

$$\begin{aligned}\frac{\partial E_{total}}{\partial \omega_1} &= \delta_{h1} x_1 \\ &= 0.009(0.1) \\ &= 0.001\end{aligned} \tag{54}$$

$$\begin{aligned}\omega_1^1 &= \omega_1^0 - \eta\frac{\partial E_{total}}{\partial \omega_1} \\ &= 0.15 - 0.5(0.001) \\ &= 0.149\end{aligned} \tag{55}$$

$$\begin{aligned}\frac{\partial E_{total}}{\partial \omega_2} &= \delta_{h1} x_2 \\ &= 0.009(0.1) \\ &= 0.001\end{aligned} \tag{56}$$

$$\begin{aligned}\omega_2^1 &= \omega_2^0 - \eta\frac{\partial E_{total}}{\partial \omega_2} \\ &= 0.2 - 0.5(0.001) \\ &= 0.199\end{aligned} \tag{57}$$

ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	b_1	b_2
0.149	0.99	0.249	0.299	0.364	0.414	0.507	0.557	0.35	0.6

Table 3: Weights after applying pattern 1

$$\begin{aligned}\delta_{h2} &= (\delta_{o1}\omega_6^0 + \delta_{o2}\omega_8^0)(out_{h2})(1 - out_{h2}) \\ &= (0.122(0.45) - 0.022(0.55))(0.6)(0.6) \\ &= 0.010\end{aligned} \tag{58}$$

$$\begin{aligned}\frac{\partial E_{total}}{\partial \omega_3} &= \delta_{h2} x_1 \\ &= 0.010(0.1) \\ &= 0.001\end{aligned} \tag{59}$$

$$\begin{aligned}\omega_3^1 &= \omega_3^0 - \eta \frac{\partial E_{total}}{\partial \omega_3} \\ &= 0.25 - 0.5(0.001) \\ &= 0.249\end{aligned} \tag{60}$$

$$\begin{aligned}\frac{\partial E_{total}}{\partial \omega_4} &= \delta_{h2} x_2 \\ &= 0.010(0.1) \\ &= 0.001\end{aligned} \tag{61}$$

$$\begin{aligned}\omega_4^1 &= \omega_4^0 - \eta \frac{\partial E_{total}}{\partial \omega_4} \\ &= 0.3 - 0.5(0.001) \\ &= 0.299\end{aligned} \tag{62}$$

3.1.2 Pattern 2

Continuing with the forward pass for pattern 2

3.2 Question 3(b)

According to Mitchell (1997) the following functions can be modeled by feed forward networks. Firstly any boolean function can be represented by using 2 layers of units (not counting the input layer). But, in the worst case, the number of hidden unit will grow exponentially with the number of inputs. Secondly any Continuous bounded function can be modeled to an arbitrary level of accuracy with a network with one hidden layer. Thirdly any arbitrary function can be approximated with a network containing 2 layers of hidden units.

3.3 Question 3(c)

The error surface for multi-layered networks contains multiple local minima. The gradient descent method used can only converge to a local minima which is not necessarily the global minimum. When the gradient decent process starts close to a local minimum the algorithm will follow the gradient to the local minimum (Mitchell, 1997).

Networks with higher dimensions may suffer less from local minima because when the network reches a local minimum in a certain dimension, the other dimentions are not necessarily at a minumum. Their weights can get the network out of some local minima (Mitchell, 1997).

The problem of local minima can also lessened by using a momentum term. As in

$$\omega_i^{j+1} = \omega_i^j - \eta \frac{\partial E_{total}}{\partial \omega_i^j} + \alpha(\omega_i^j - \omega_i^{j-1}) \tag{63}$$

where alpha is the momentum constant and this depends on the previous change in the weight. This way the algorithm can get out of a relatively small local minimum (Mitchell, 1997).

Another method is simulated annealing (Rojas, 1996);

3.4 Question 3(d)

A discussion of the identity function can be found in (Mitchell, 1997, p106) at $http://www.cs.ubbcluj.ro/ \sim gabis/ml/ml-books/McGrawHill-MachineLearning-TomMitchell.pdf$

Using a 8 X 3 X 8 network we want to represent the identity function. In other words when any input node is given an high input we want the corresponding output node to be high. Thus if input 2 is high and all the other nodes are low (01000000), output 2 needs to be high and all other outputs low (01000000). The problem is that there is not enough nodes in the hidden layer to simply forward the value from the input node to the corresponding output node.

When we attempt to train such a network the network finds a representation in the hidden layer to represent all 8 inputs. The network uses a binary representation where the input (00100000) will be represented as (011) in the hidden units. This will then be output as (00100000). This way the network learn a new representation that it was not explicitly told.

The values from the book can be found in table 4

3.5 Question 3(e)

The sigmoid function is defined as

$$\sigma(y) = \frac{1}{1 + e^{-y}} \tag{64}$$

References

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Input	h_1	h_2	h_3	*Output*
10000000	0.89	0.04	0.08	10000000
01000000	0.15	0.99	0.99	01000000
00100000	0.01	0.97	0.27	00100000
00010000	0.99	0.97	0.71	00010000
00001000	0.03	0.05	0.02	00001000
00000100	0.01	0.11	0.88	00000100
00000010	0.80	0.01	0.98	00000010
00000001	0.60	0.94	0.01	00000001

Table 4: Hidden unit representation

Nils J Nilsson. (1998). *Introduction to Machine Learning.* Retrieved from `http://robotics.stanford.edu/people/nilsson/MLBOOK.pdf`

Rojas, R. (1996). *Neural Networks A Systematic Introduction.* Springer-Verlag Springer-Verlag. Retrieved from `https://page.mi.fu-berlin.de/rojas/neural/neuron.pdf`

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. Retrieved from `http://www.cs.toronto.edu/~hinton/absps/naturebp.pdf` doi: 10.1038/323533a0