Tutorial 10:
Neural Networks

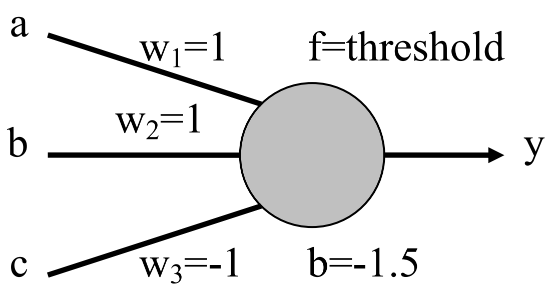
Question 1

(example answers)

a) W1 = 2, w2 = 2, w0 = b = -3

b) a^b^!c (! = NOT, ^ = AND)

Use a single threshold unit
with 3 inputs: a, b and c.
The given function can be modelled (for 0/1 binary data anyway)
by using the following set of weights:
1, 1 and -1 on a, b, and c inputs respectively, with a -1.5 bias weight.



Question 2

X1=[0 0];
hidden unit activation values ui (i=1,2)
u1 = 0*-4 + 0*4 +2 = 2; h1 = 1/(1+exp(-u1)) ~= 0.88
u2 = 0*-5 + 0*5 -3 = -3; h2 = 1/(1+exp(-u2)) ~= 0.047
output unit activation value z
z = 0.88*-4 + 0.047*3 +2 = -1.38; y (output) = 1/(1+exp(-z)) ~= 0.20
This is <0.5, so counts as class A.

X2=[0 1]
u1 = 0*-4 + 1*4 +2 = 6; h1 = 1/(1+exp(-u1)) ~= 0.998
u2 = 0*-5 + 1*5 -3 = 2; h2 = 1/(1+exp(-u2)) ~= 0.88
z = 0.998*-4 + 0.88*3 +2 = 0.648; y (output) = 1/(1+exp(-z)) ~= 0.657
This is >0.5, so counts as class B.

X3=[1 0]
u1 = 1*-4 + 0*4 +2 = -2; h1 = 1/(1+exp(-u1)) ~= 0.12
u2 = 1*-5 + 0*5 -3 = -8; h2 = 1/(1+exp(-u2)) ~= 0.00034
z ~= 0.12*-4 +2 = 1.52; y (output) = 1/(1+exp(-z)) ~= 0.821
This is >0.5, so counts as class B.

X4=[1 1]
u1 = 1*-4 + 1*4 +2 = 2

u2 = 1*-5 + 1*5 -3 = -3
-> same as X1
z = 0.88*-4 + 0.047*3 +2 = -1.38; y (output) = 1/(1+exp(-z)) ~= 0.20
This is <0.5, so counts as class A.

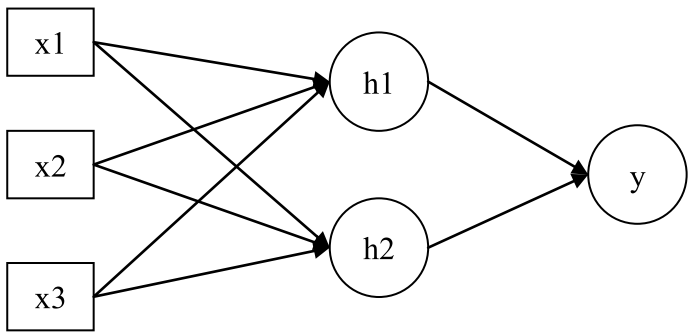
Question 3

Classification:
Networks with a single layer of weights can only solve linearly separable problems by implementing a straight line decision boundary in input space. e.g. OR, AND.
Having a hidden layer allows the network to implement non-linear decision boundaries in input space, thus potentially modelling any classification function (provided there are enough hidden units).

Regression:
The contours in input space which will produce equal output (e.g. all producing 0.6) will also be able to become non-linear when hidden units are used. With only one layer of weights, they would all have to be straight lines. The ability of a 2-layer (of weights) NN to model any input->output function has also been shown (for an arbitrary # of hidden units) by e.g.
K. Hornik, M. Stinchcombe, H. White (1989)
Multilayer feedforward networks are universal approximators
Neural Networks Volume 2 , Issue 5 (1989) pp 359 - 366
(available through library - but pretty mathematical - you'd want to know what Borel-measurable means to follow all of it.).

Question 4

This function is not linearly separable, so you need to use a multi-layer network to solve it. We looked at the 2 input XOR function in class, this is pretty much the same, just with 3 inputs. There are several different ways that you can approach this problem, and many more combinations of weights and biases that will work. Here, I'll use an approach similar to the one we looked at in the lecture. There, we had 2 hidden units, one calculating OR and the other calculating AND. To match this truth table, however, the second hidden unit will calculate when at least 2 of x1, x2, and x3 are active.



H1 is going to implement (x1 or x2 or x3)
H2 is going to implement ((x1 and x2) OR (x1 and x3) OR (x2 and x3))
Y is going to implement (h1 and (not h2))

H1:
Basic OR function, Output is 1 if at least one of the inputs is active
Equal weights (1.0 each), with a bias value of -0.5
W(x1 to h1) = 1.0
W(x2 to h1) = 1.0
W(x3 to h1) = 1.0

B(h1) = -0.5

H2:
Output is 1 when at least 2 of the 3 inputs are active
Equal weights (1.0 each), with a bias value of -1.5
W(x1 to h2) = 1.0
W(x2 to h2) = 1.0
W(x3 to h2) = 1.0
B(h2) = -1.5

Y:
Output is 1 if h1 is active and h2 is not active
W(h1 to y) = 1.0
W(h2 to y = -1.0
B(y) = -0.5

Question 5

1. Feedforward activations through network
2. Propagate error back through network
3. Update weights according to blame attributed to each weight