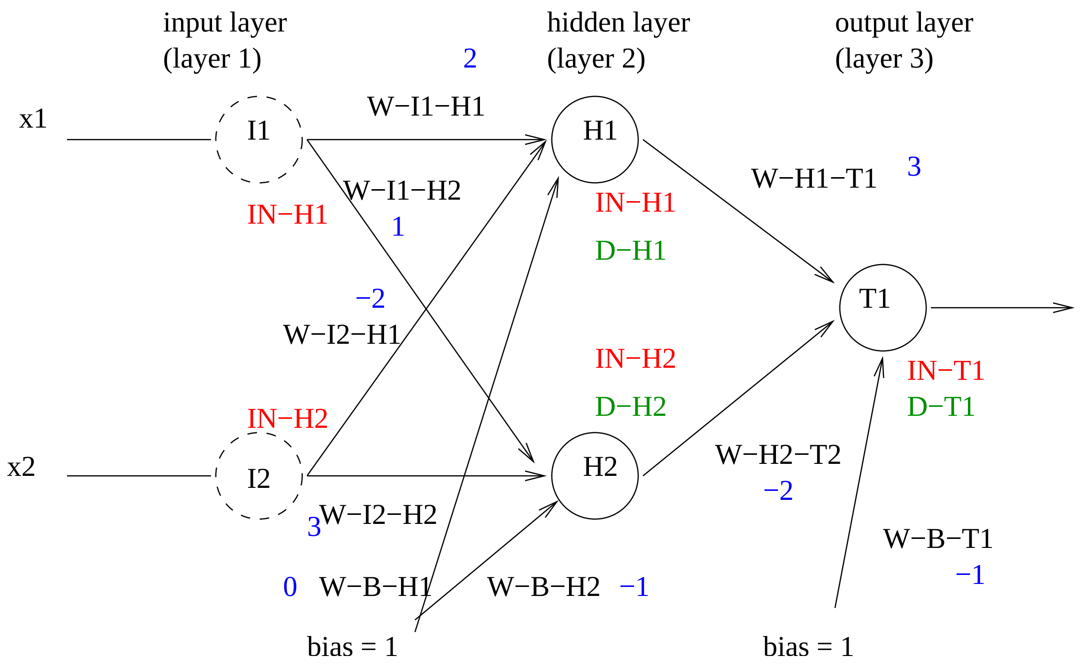
CS4811 Neural Network Training Example

Consider the following network. It has two inputs (two entries in each input vector), one hidden layer with two neurons and the output layer with a single neuron. Each neuron has a bias input to allow threshold values other than 0.



Each neuron is connected to all the neurons in the next layer, there are no back connections (this is a *feedforward network*).

Each neuron has an `IN` value that represents the weighted sum of its inputs. The neurons in the input layer simply output their input, they don't have weights coming in. That is why they are drawn as dashed lines. Each neuron also has a `D` value that represents the delta value that will be used for back propagation. Notice that the nodes in the input layer do not have `D`s associated because they don't have incoming weights.

Let's assume that this network is going to learn the NXOR function. So there are 4 examples. For simplicity we will not be showing the bias in the examples, we will always assume it is 1 for all the neurons. The training examples are the following:

x1 = 1	x2 = 0	desired=0
x1 = 0	x2 = 0	desired=1
x1 = 0	x2 = 1	desired=0
x1 = 1	x2 = 1	desired=1

We will first initialize the weights to random values:

```
W-I1-H1 = 2                     W-I1-H2 = 1
W-I2-H1 = -2                    W-I2-H2 = 3
W-B-H1 = 0                      W-B-H2 = -1
W-H1-T1 = 3
W-H2-T1 = -2
W-B-T1 = -1
```

We will now do a complete pass with the first training example `x1 = 1, x2 = 0` and the desired output is `0`. We first compute the output of all the neurons. For that, we will use the *sigmoid function* rather than the threshold function because it is differentiable. If f is the sigmoid function, then its differential is $f(1-f)$. The sigmoid function is defined as follows:

$$\frac{1}{1+e^{-x}}$$

The input layer simply transfers to the input to the output:

`SUM-I1` = $in_{i1} = a_{i1}$ = $x1$
`SUM-I2` = $in_{i2} = a_{i2}$ = $x2$

The other neurons compute a weighted sum of their inputs:

`SUM-H1` = $\sum_j W_{j,i} a_j$ = $x1\times$ `W-I1-H1` + $x2\times$ `W-I2-H1` + `bias` $\times$ `W-B-H1`= $1\times 2+0\times -2+1\times 0=2$

`IN-H1` = sigmoid (`SUM-H1`) = sigmoid (2) = 0.880797.

Similarly, `SUM-H2` = 0, `IN-H2` = 0.5, and `SUM-T1` = 0.642391, and `IN-T1` = 0.655294.

Next we will compute the delta values. We start with the output layer:

`D-T1` = (desired - `IN-T1`) $\times$ `IN-T1` $\times$ (1- `IN-T1`) = -0.148020.

We continue with the two nodes in the hidden layer:

`D-H1` = `IN-H1` $\times$ (1 - `IN-H1`) $\times$ `D-T1` $\times$ `W-H1-T1` = -0.0446624.

`D-H2` = `IN-H2` $\times$ (1 - `IN-H2`) $\times$ `D-T1` $\times$ `W-H2-T1` = 0.074010.

Now we have all the values we need, we will perform backpropagation and update the weights. We will use a learning constant c of 1. If another value is used, it multiplies the term after the sum:

`W-H1-T1` = `W-H1-T1` + $c \times$ `IN-H1` $\times$ `D-T1`
= 3 + 1 $\times$ 0.880797 $\times$ -0.148020
= 2.869624.

`W-H2-T1` = `W-H2-T1` + $c \times$ `IN-H2` $\times$ `D-T1`
= -2.074010.

`W-B-T1` = `W-B-T1` + $c \times 1 \times$ `D-T1`
= -1.148020.

`W-I1-H1` = `W-I1-H1` + $c \times x1 \times$ `D-H1`
= 1.953376.

`W-I2-H1` = `W-I2-H1` + $c \times x2 \times$ `D-H1`
= -2.000000.

`W-B-H1` = `W-B-H1` + $c \times 1 \times$ `D-H1`
= -0.046624.

`W-I1-H2` = 1.074010
`W-I2-H2` = 3.000000
`W-B-H2` = -0.925990.

The above are the weights after the first example is processed. A similar procedure is followed several times for each example until the network converges or an iteration limit is reached. To test the network, we go back to using the threshold function (not the sigmoid).