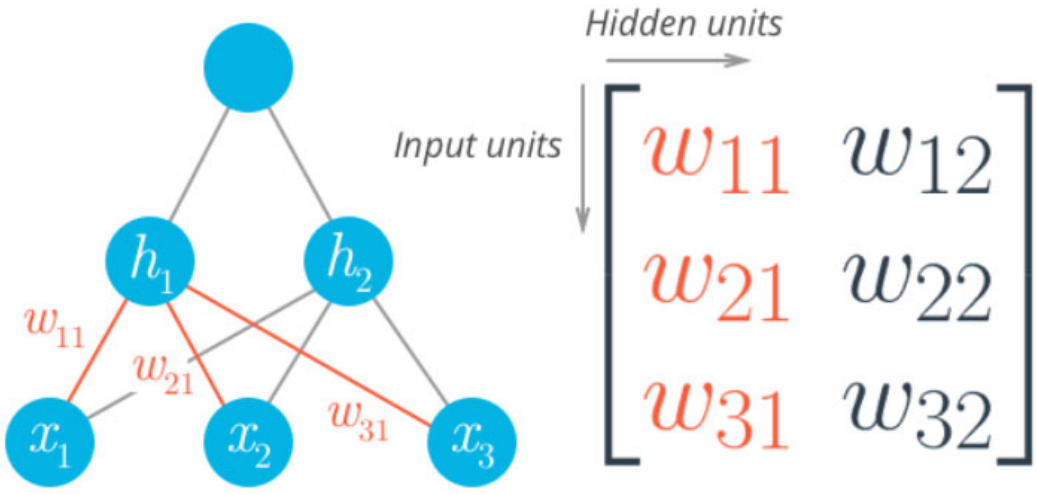
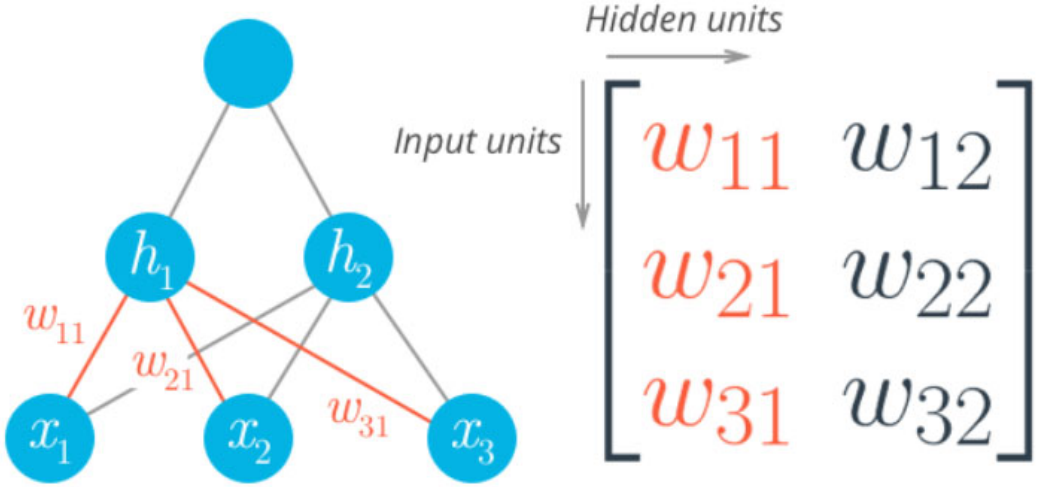
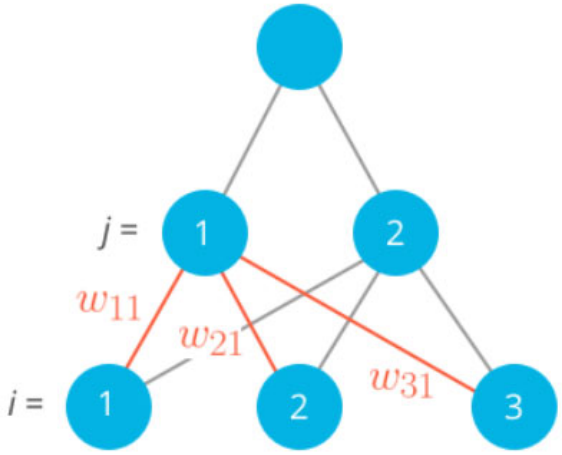

Hidden units
Input units
h_1
h_2
x_1
x_2
x_3
w_{11}
w_{21}
w_{31}
w_{11} w_{12}
w_{21} w_{22}
w_{31} w_{32}


Hidden units
Input units
h_1
h_2
x_1
x_2
x_3
w_{11}
w_{21}
w_{31}
w_{11} w_{12}
w_{21} w_{22}
w_{31} w_{32}

$$h_j = \sum_i w_{ij} x_i$$



$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

Calculating the input to the first hidden unit with the first column of the weights matrix.

$$h_1 = x_1 w_{11} + x_2 w_{21} + x_3 w_{31}$$

Below is the implementation of a forward pass through a 4x3x2 network, with sigmoid activation functions for both layers. Below are the steps

1.Calculate the input to the hidden layer.

2.Calculate the hidden layer output.

3.Calculate the input to the output layer.

4.Calculate the output of the network.

In [1]:
```
import numpy as np

def sigmoid(x):
    """
    Calculate sigmoid
    """
    return 1/(1+np.exp(-x))

# Network size
N_input = 4
N_hidden = 3
N_output = 2

np.random.seed(42)
# Make some fake data
X = np.random.randn(4)

weights_input_to_hidden = np.random.normal(0, scale=0.1, size=(N_input, N_hidden))
weights_hidden_to_output = np.random.normal(0, scale=0.1, size=(N_hidden, N_output))

# Make a forward pass through the network

hidden_layer_in = None
hidden_layer_out = None

hidden_layer_in = np.dot(X,weights_input_to_hidden)
#hidden_layer_in = np.matmul(X,weights_input_to_hidden)
hidden_layer_out = sigmoid(hidden_layer_in)


print('Hidden-layer Output:')
print(hidden_layer_out)

output_layer_in = None
output_layer_out = None

output_layer_in = np.dot(hidden_layer_out,weights_hidden_to_output)
output_layer_out =sigmoid(output_layer_in)


print('Output-layer Output:')
print(output_layer_out)
```

```
Hidden-layer Output:
[0.41492192 0.42604313 0.5002434 ]
Output-layer Output:
[0.49815196 0.48539772]
```

In [2]:
```
weights_input_to_hidden = np.random.normal(0, scale=0.1, size=(4, 3))
```

In [3]:
```
X
```

Out[3]:
```
array([ 0.49671415, -0.1382643 ,  0.64768854,  1.52302986])
```

In [4]:
```
weights_input_to_hidden
```

Out[4]:
```
array([[ 0.00675282, -0.14247482, -0.05443827],
       [ 0.01109226, -0.11509936,  0.0375698 ],
       [-0.06006387, -0.02916937, -0.06017066],
       [ 0.18522782, -0.00134972, -0.10577109]])
```

In [5]:
```
np.matmul(X,weights_input_to_hidden)
```

Out[5]:
```
array([ 0.24502538, -0.07580346, -0.2322992 ])
```

In [6]:
```
np.dot(X,weights_input_to_hidden)
```

Out[6]:
```
array([ 0.24502538, -0.07580346, -0.2322992 ])
```