



Weights matrix for 3 input units and 2 hidden units

Be sure to compare the matrix above with the diagram shown before it so you can see where the different weights in the network end up in the matrix.

To initialize these weights in NumPy, we have to provide the shape of the matrix. If `features` is a 2D array containing the input data:

```
# Number of records and input units
n_records, n_inputs = features.shape
# Number of hidden units
n_hidden = 2
weights_input_to_hidden = np.random.normal(0, n_inputs**-0.5, size=(n_inputs, n_hidden))
```

This creates a 2D array (i.e. a matrix) named `weights_input_to_hidden` with dimensions `n_inputs` by `n_hidden`. Remember how the input to a hidden unit is the sum of all the inputs multiplied by the hidden unit's weights. So for each hidden layer unit, h_j , we need to calculate the following:

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

To do that, we now need to use [matrix multiplication](#). If your linear algebra is rusty, I suggest taking a look at the suggested resources in the prerequisites section. For this part though, you'll only need to know how to multiply a matrix with a vector.