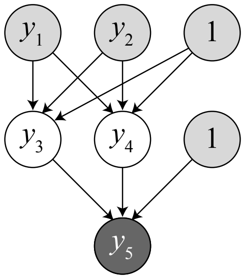
Backpropagation in a simple network

The binary XOR function is given by the following truth table:

y_1	y_2	$y_1 \otimes y_2$
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

It can be computed by a simple two-layer network with the following structure:



Note that both the input and hidden layer of the network have bias nodes. The output is computed by

$$\begin{aligned}
y_3 = f(x_3) &= f(w_{03} + w_{13}\, y_1 + w_{23}\, y_2) \\
y_4 = f(x_4) &= f(w_{04} + w_{14}\, y_1 + w_{24}\, y_2) \\
y_5 = f(x_5) &= f(w_{05} + w_{35}\, y_3 + w_{45}\, y_4)
\end{aligned}$$

where w_{ij} is the weight from node i to node j and w_{0j} is the weight from a bias node.

Download the `xor_nnet.py` file from the course webpage and complete the steps marked **`# TODO`** to implement the back-propagation algorithm described in the class handout for this simple network. If you keep the α parameter and tolerance specified in the file, your implementation should converge in 61 iterations.

Turn in the modified file on Moodle. Please make sure you write your name in a comment at the top of the file.