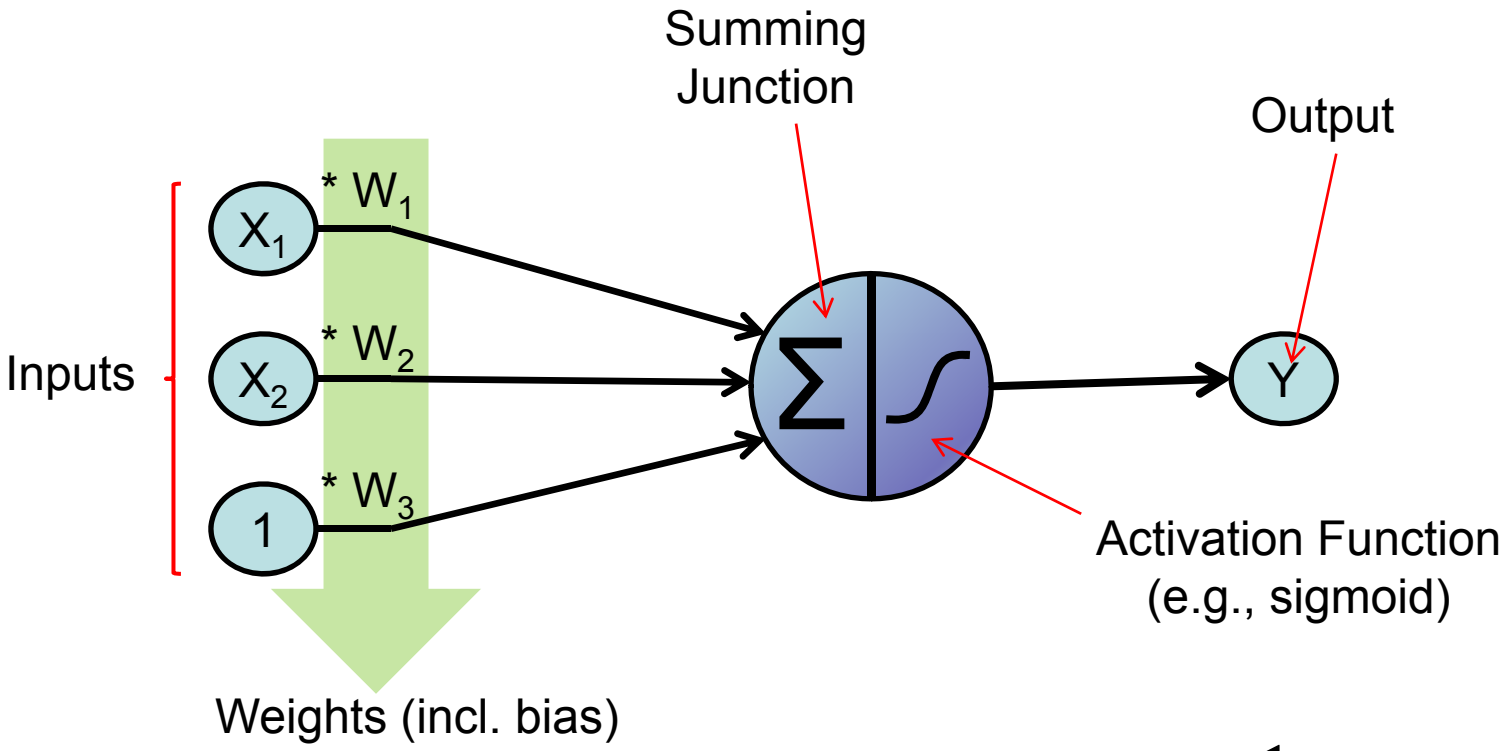
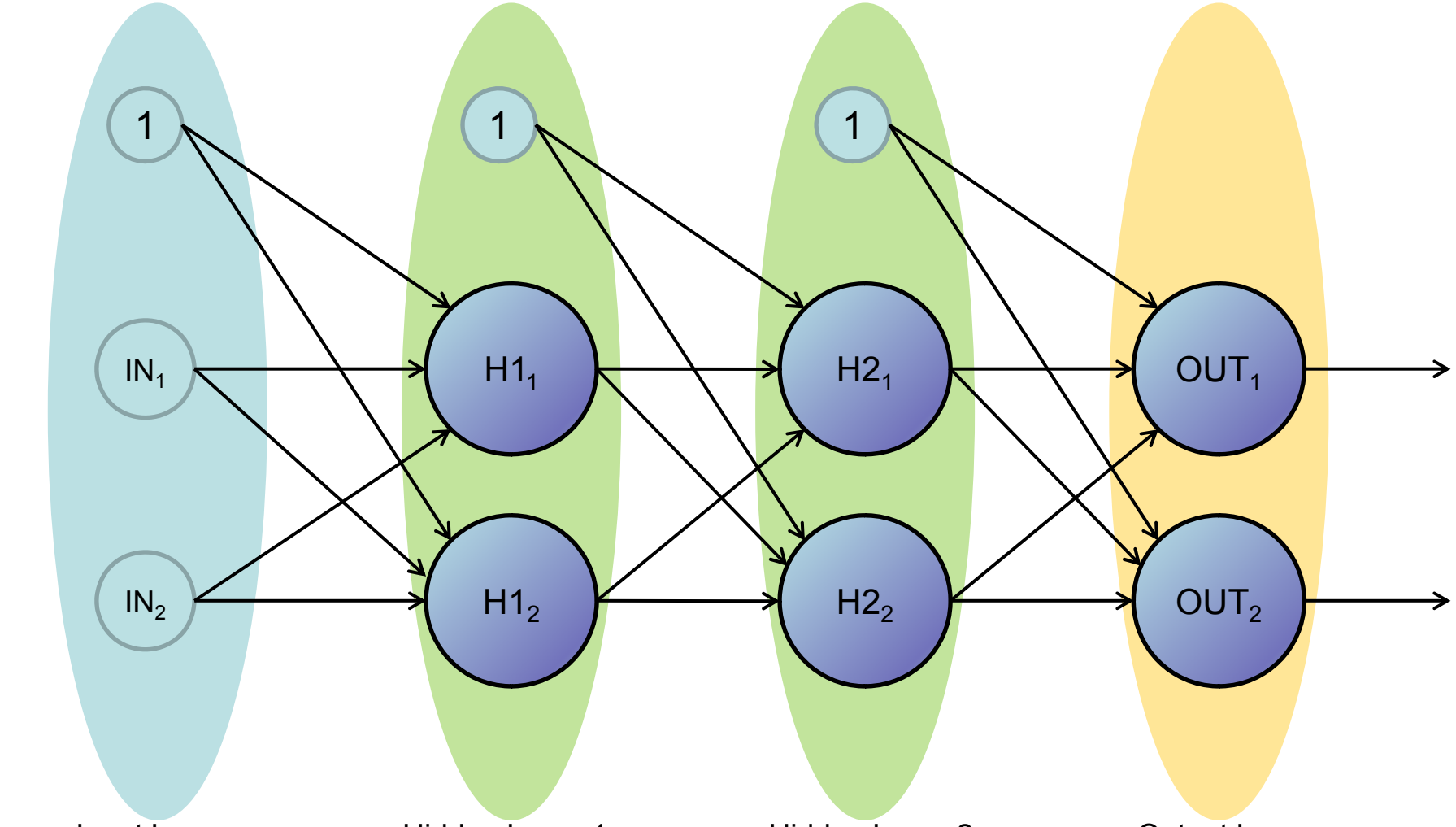
Artificial Neuron



$$Y = \frac{1}{1 + e^{-a*\left(X_1 * W_1 + X_2 * W_2 + W_3\right)}}$$

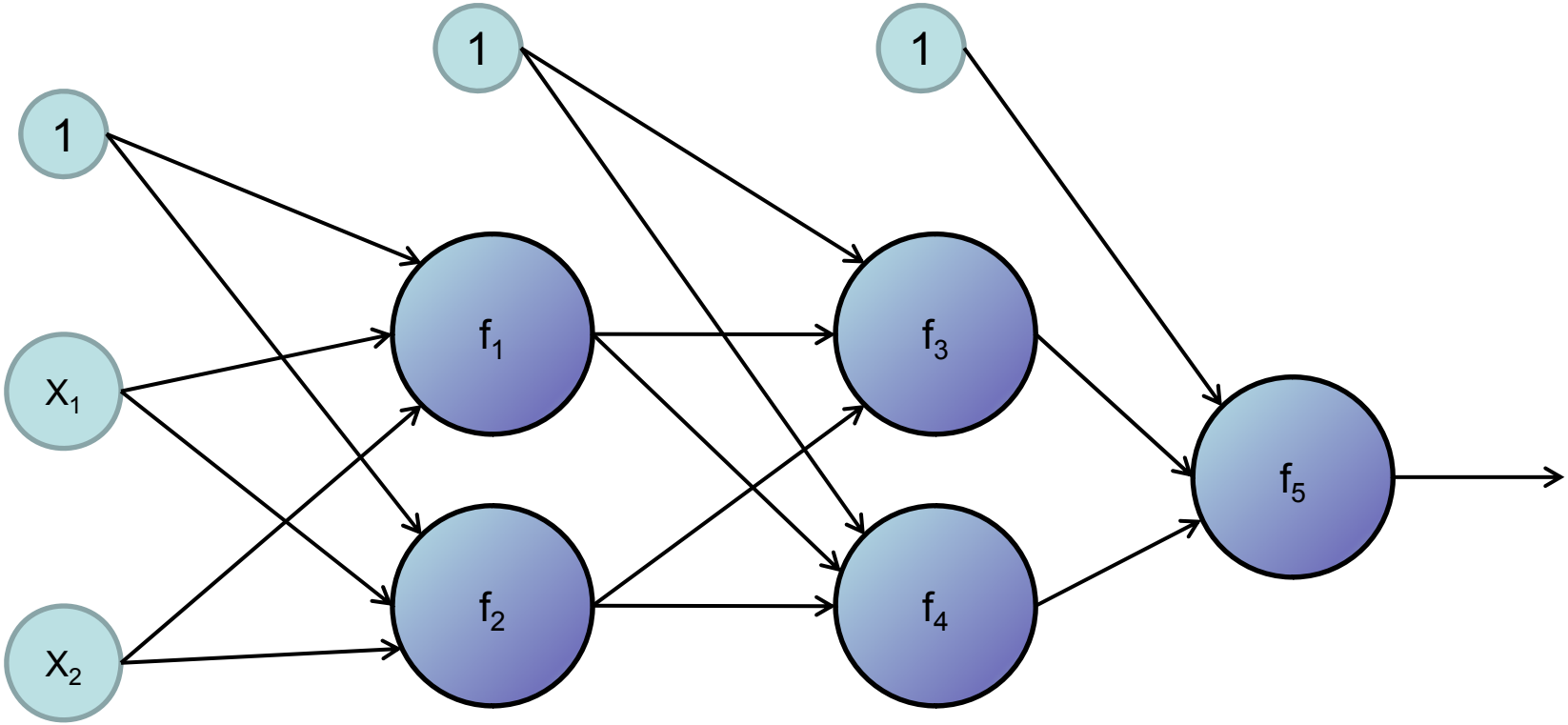
Feedforward Neural Network



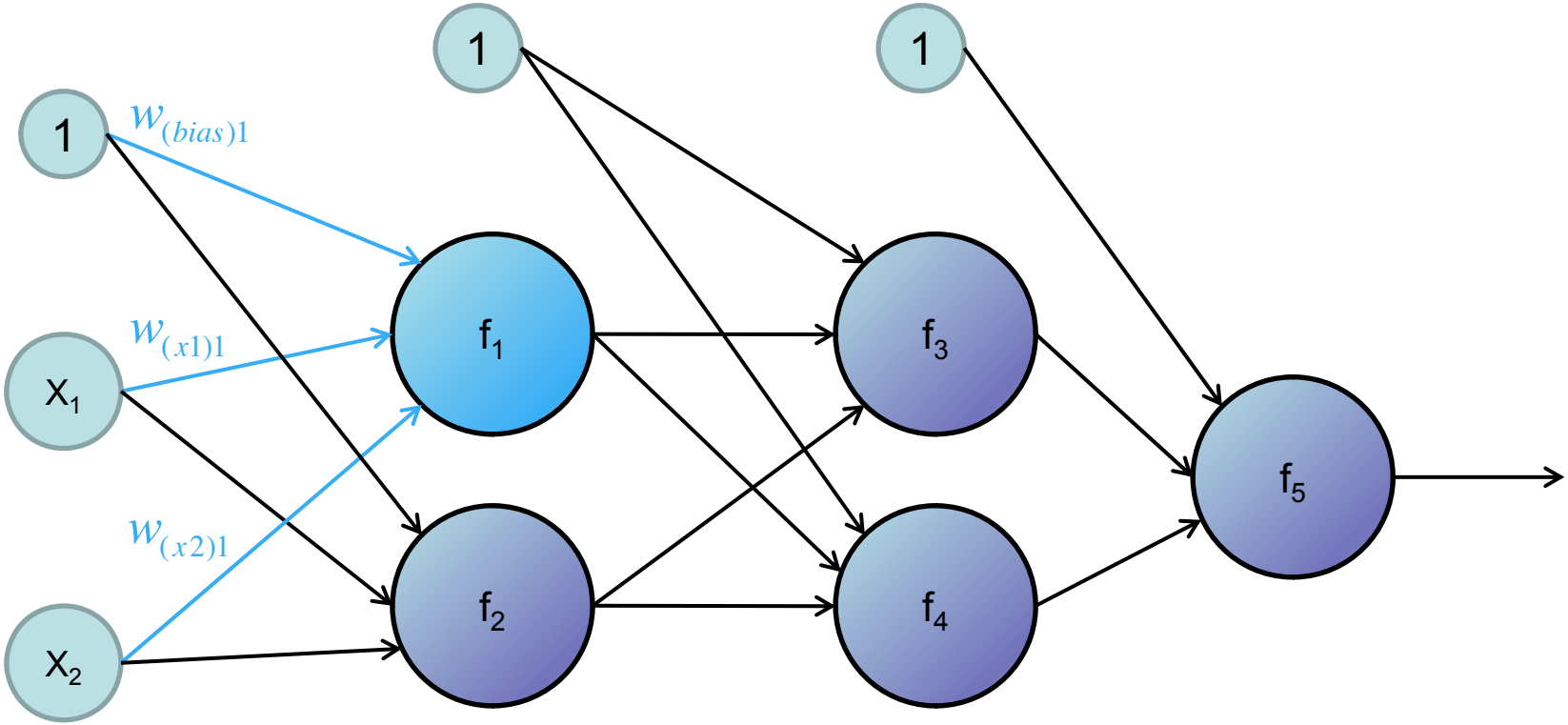
Back-propagation Algorithm (Online Updating)

Step by Step

Pattern p is presented

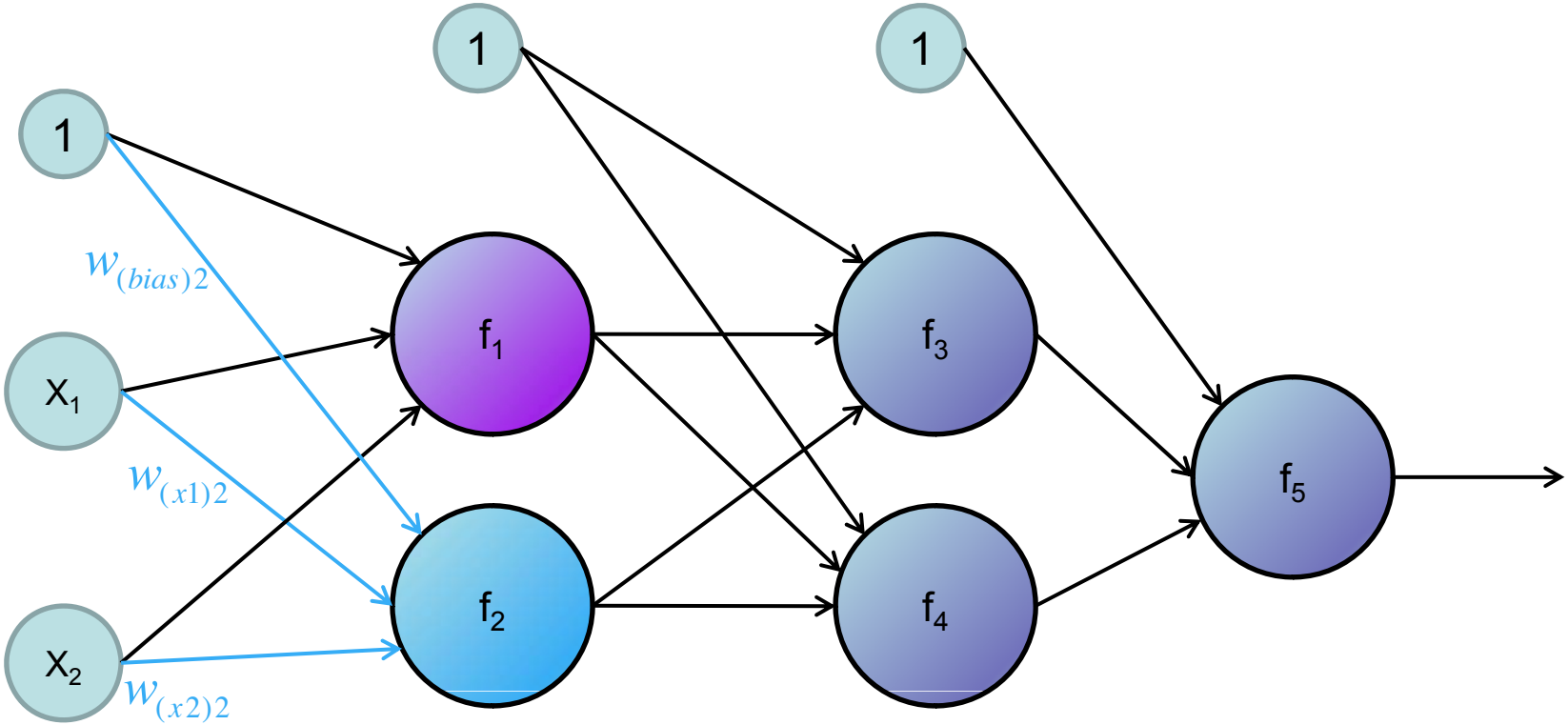


Forward Pass



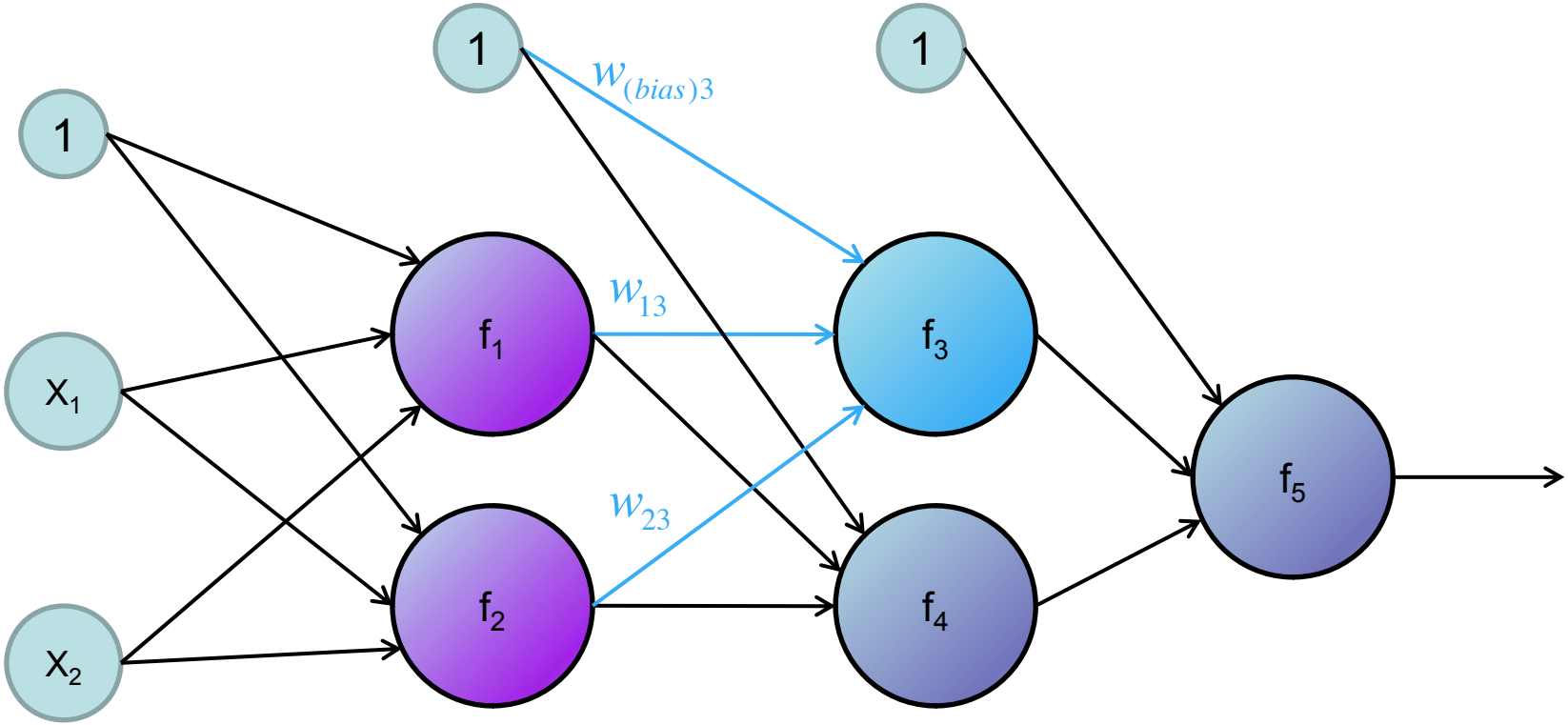
$$y_1 = f_1(w_{(bias)1} + w_{(x1)1} \cdot x_1 + w_{(x2)1} \cdot x_2)$$

Forward Pass



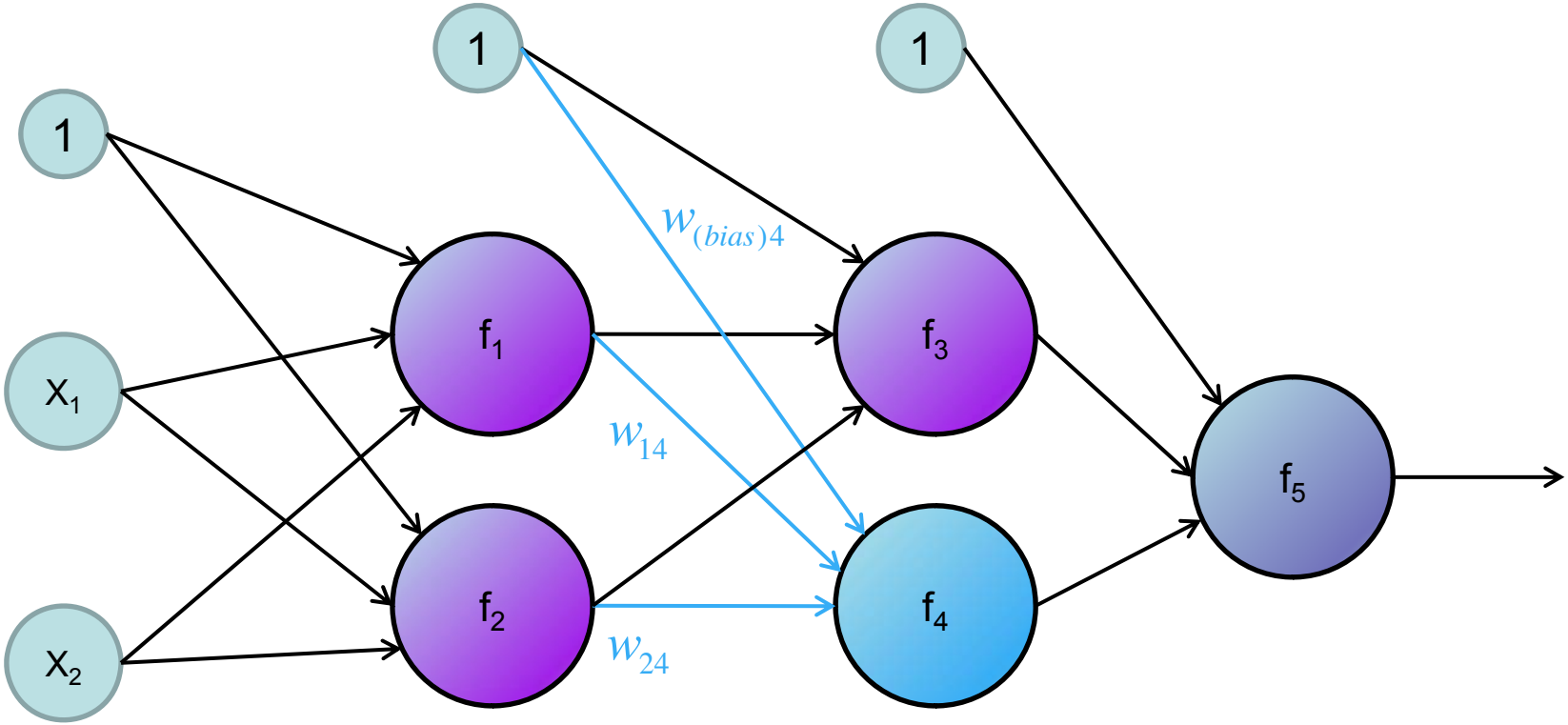
$$y_2 = f_2(w_{(bias)2} + w_{(x1)2} \cdot x_1 + w_{(x2)2} \cdot x_2)$$

Forward Pass



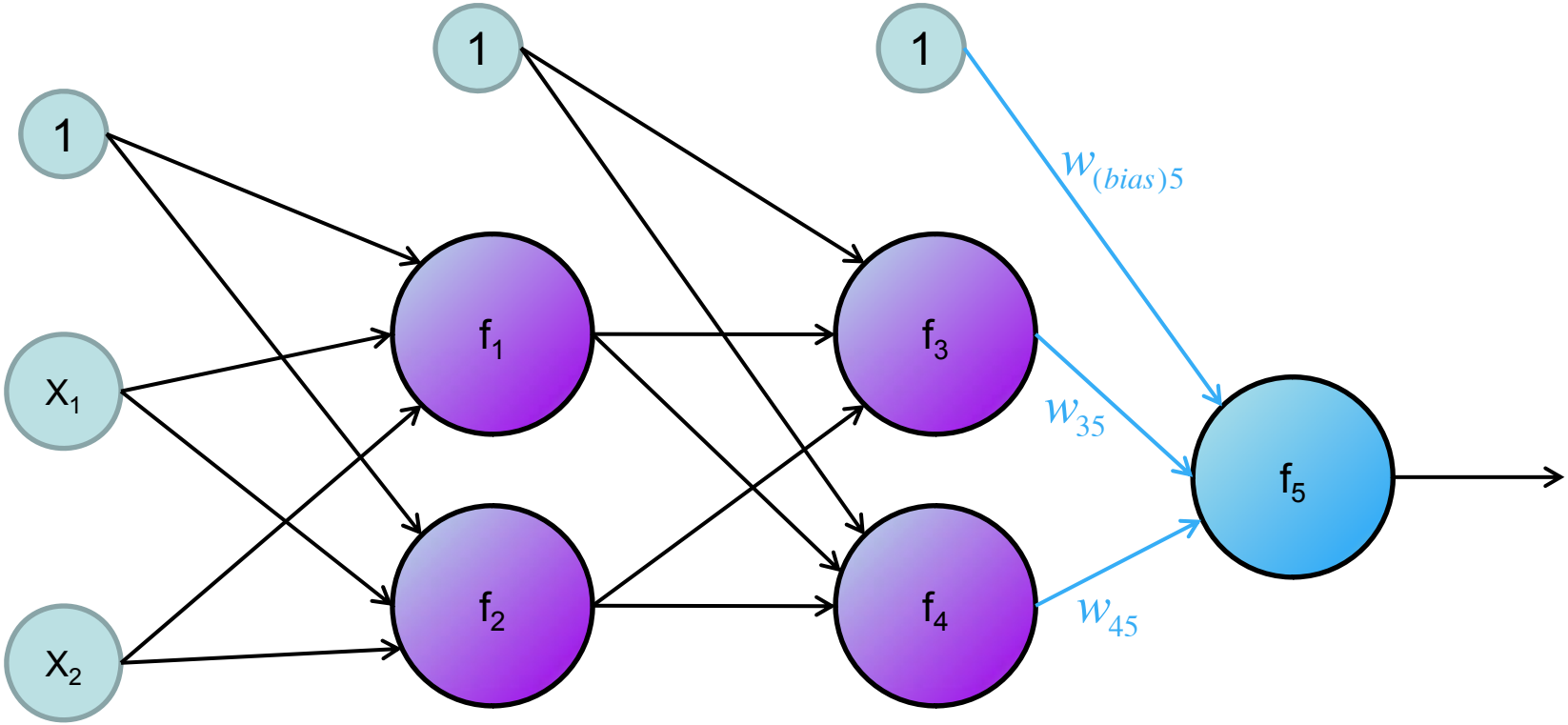
$$y_3 = f_3(w_{(bias)3} + w_{13} \cdot y_1 + w_{23} \cdot y_2)$$

Forward Pass



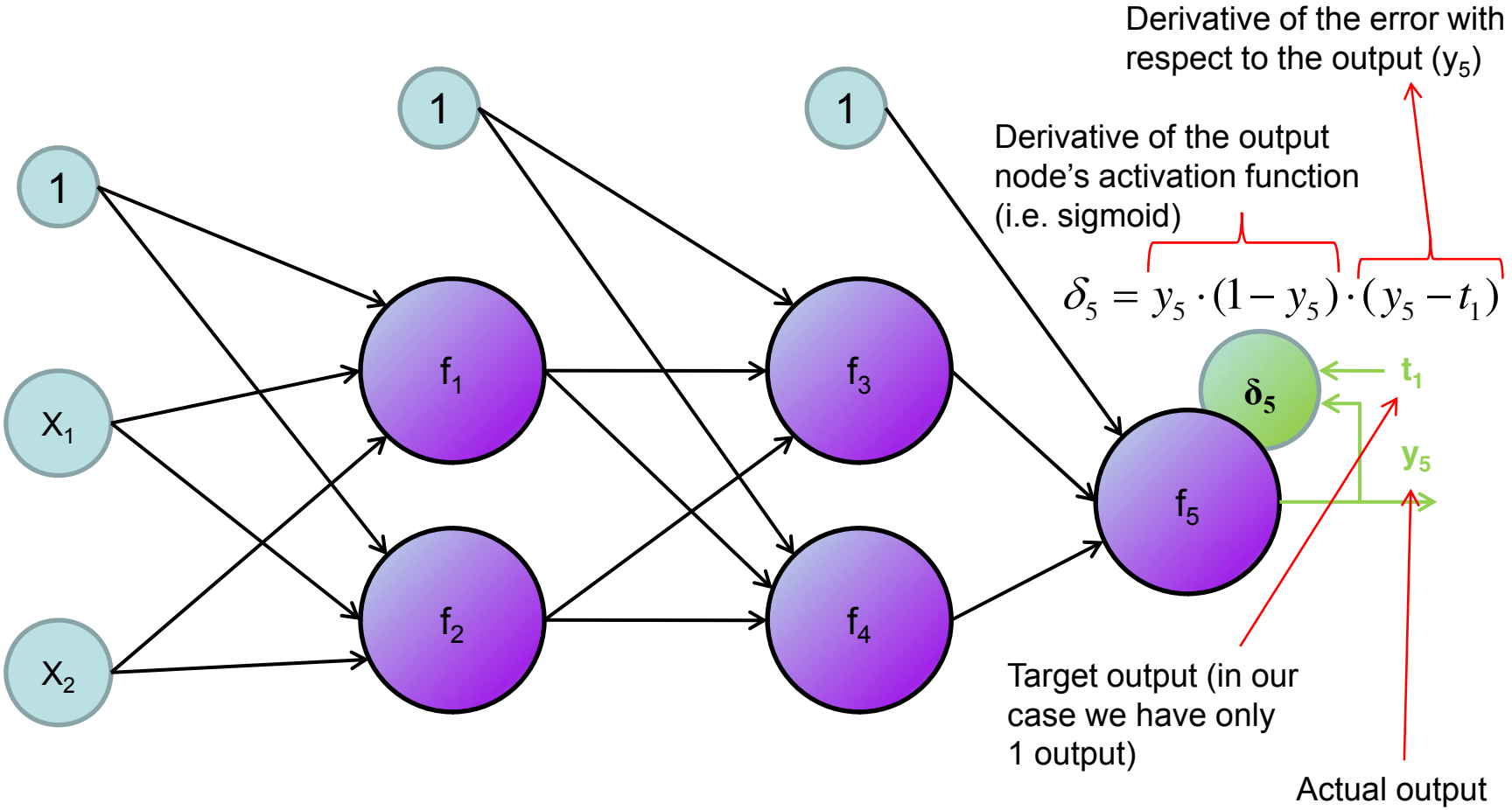
$$y_4 = f_4(w_{(bias)4} + w_{14} \cdot y_1 + w_{24} \cdot y_2)$$

Forward Pass



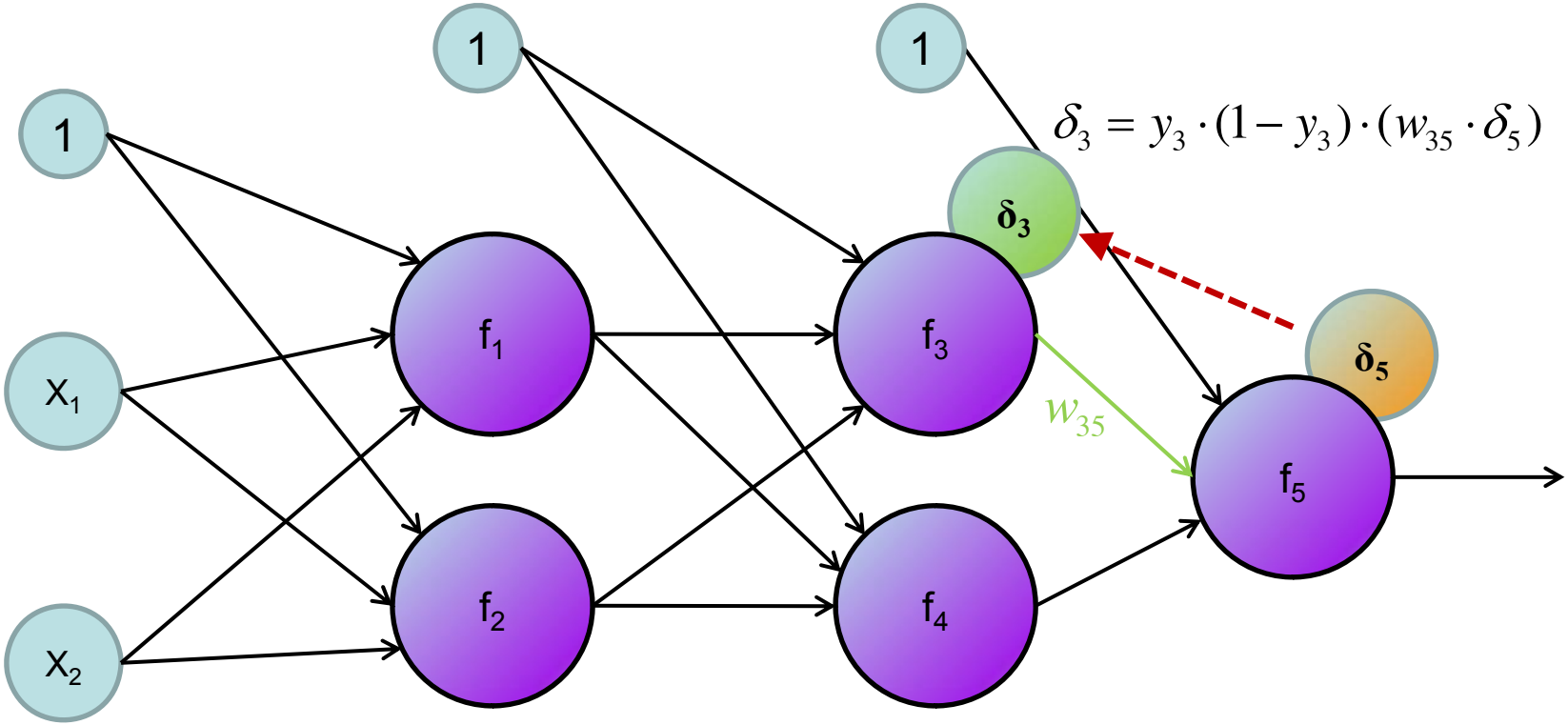
$$y_5 = f_5(w_{(bias)5} + w_{35} \cdot y_3 + w_{45} \cdot y_4)$$

Backward Pass - Stage 1

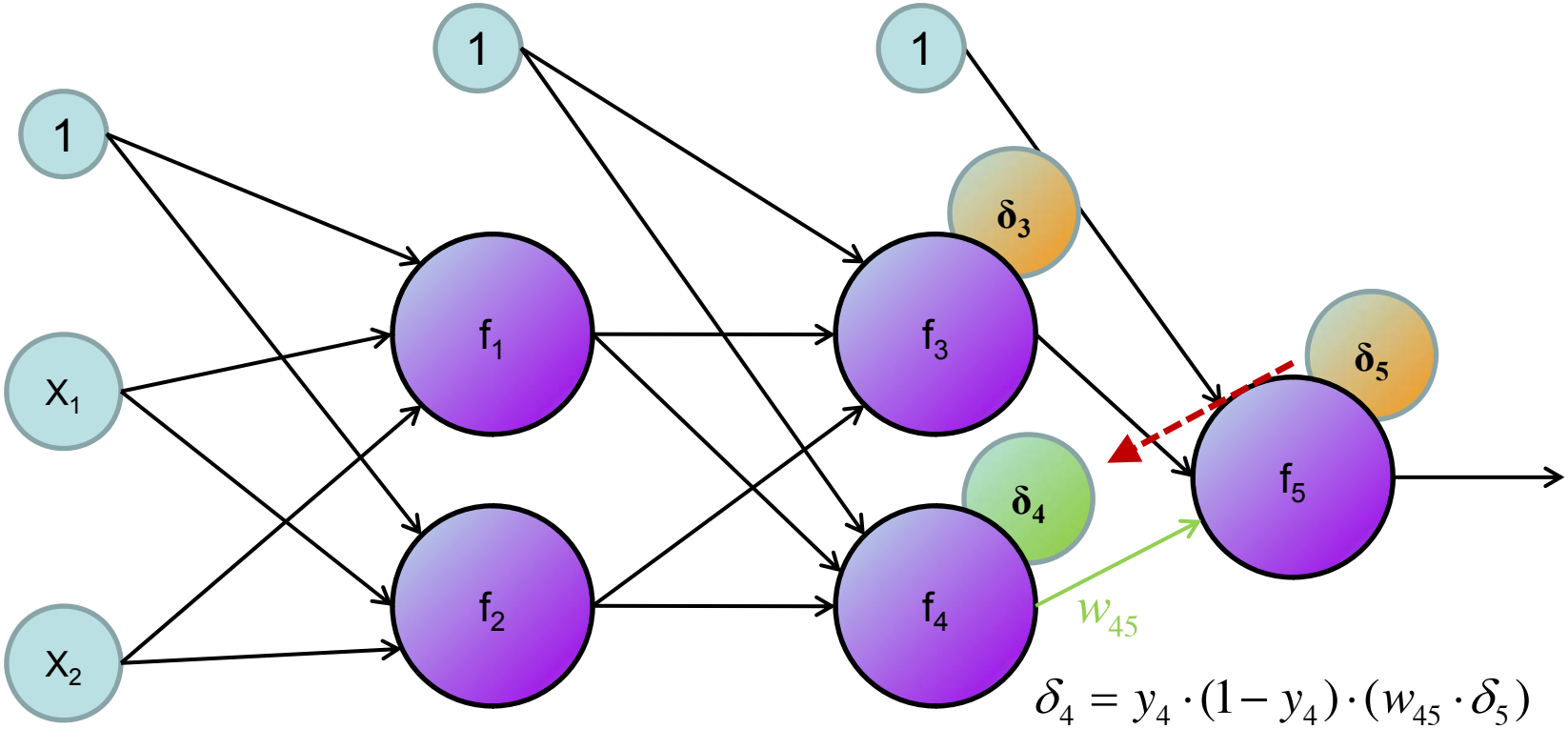


Assuming we are minimising the sum-of-squares error function and that each neuron has a sigmoid activation function with slope α=1

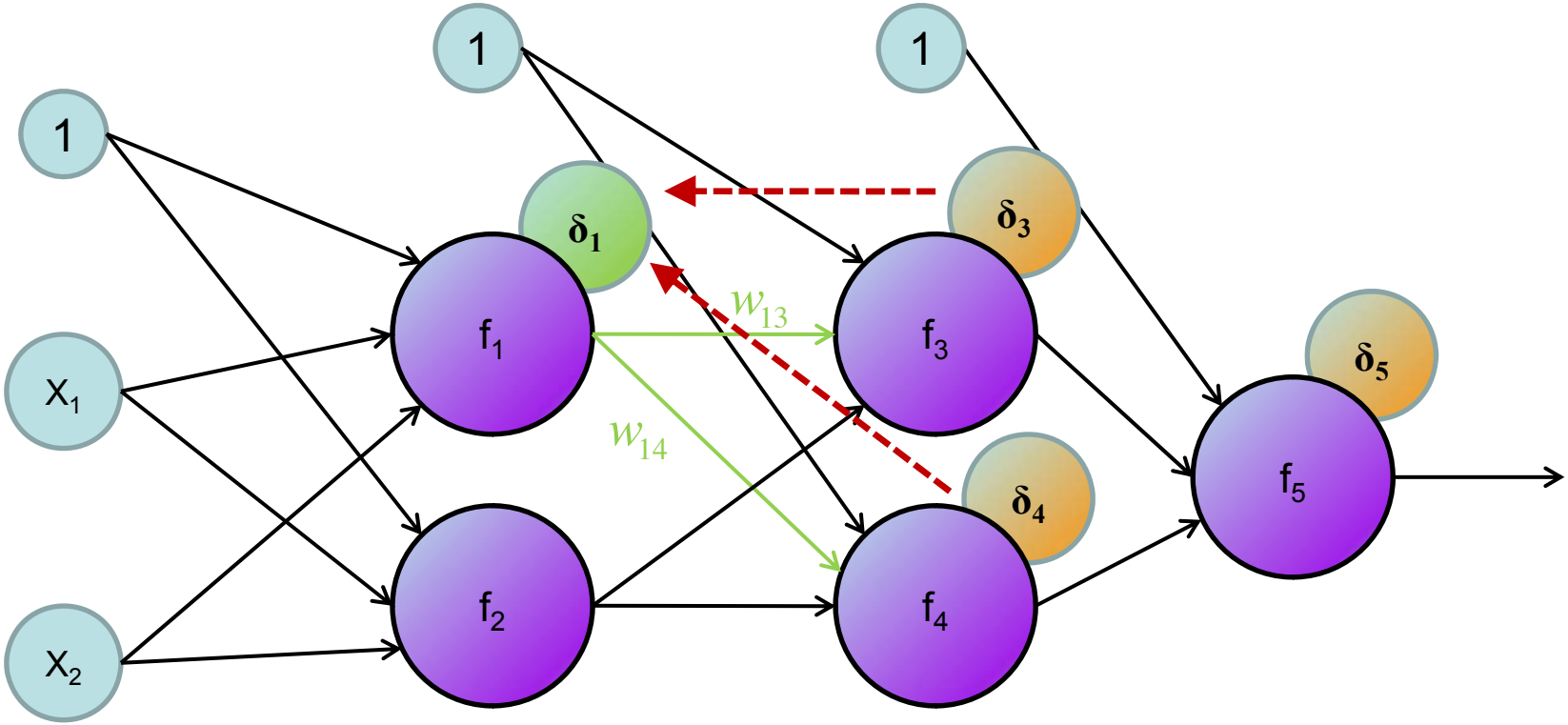
Backward Pass - Stage 1



Backward Pass - Stage 1

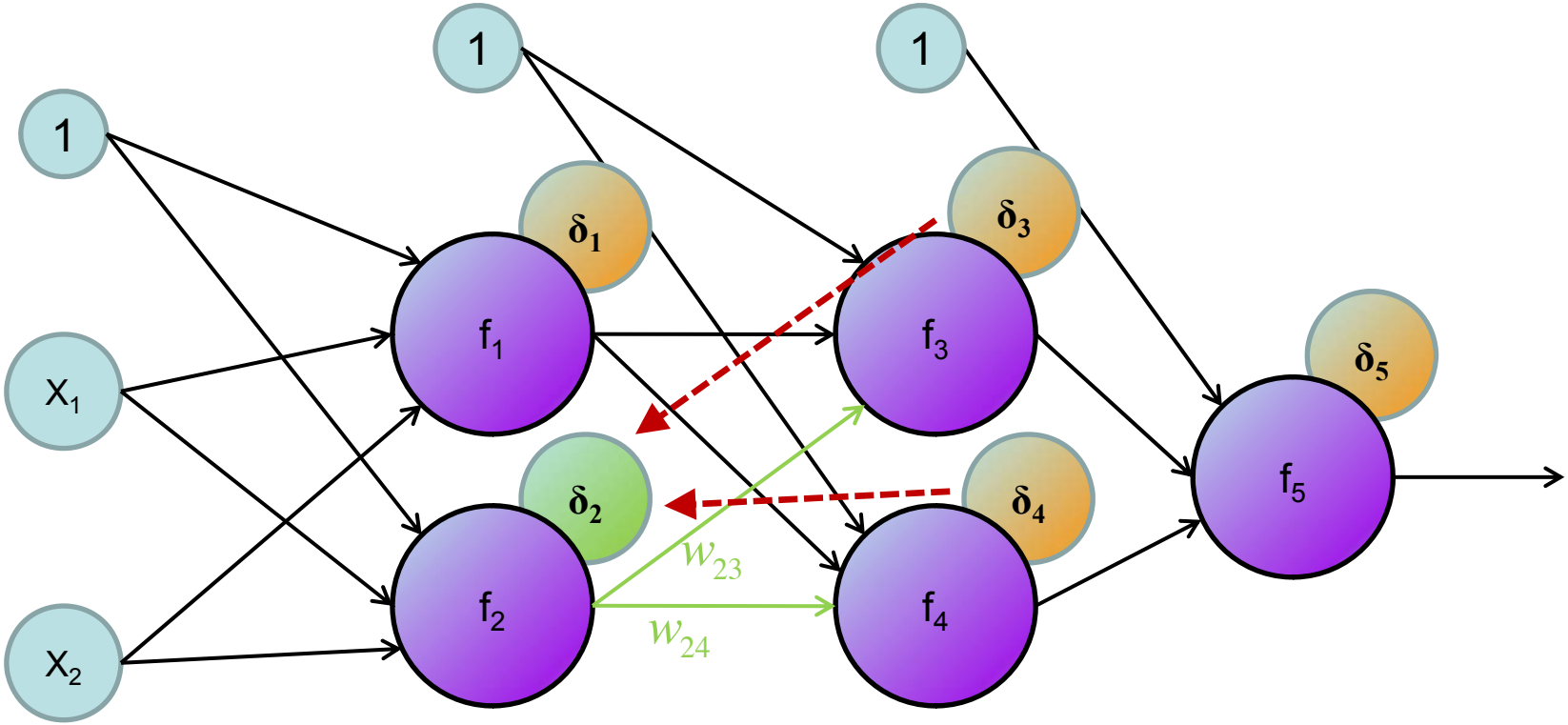


Backward Pass - Stage 1



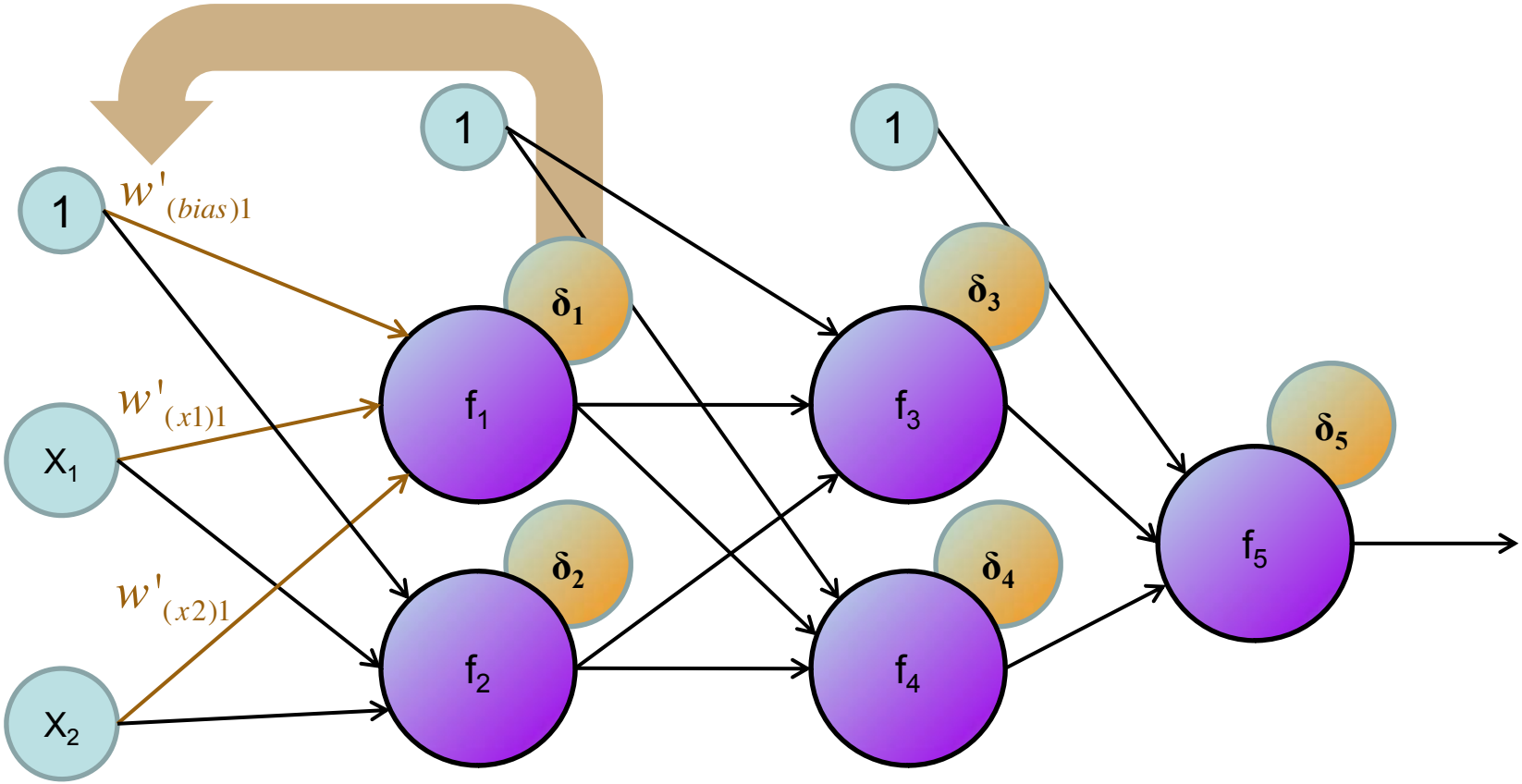
$$\delta_1 = y_1 \cdot (1 - y_1) \cdot (w_{13} \cdot \delta_3 + w_{14} \cdot \delta_4)$$

Backward Pass - Stage 1



$$\delta_2 = y_2 \cdot (1 - y_2) \cdot (w_{23} \cdot \delta_3 + w_{24} \cdot \delta_4)$$

Backward Pass - Stage 2

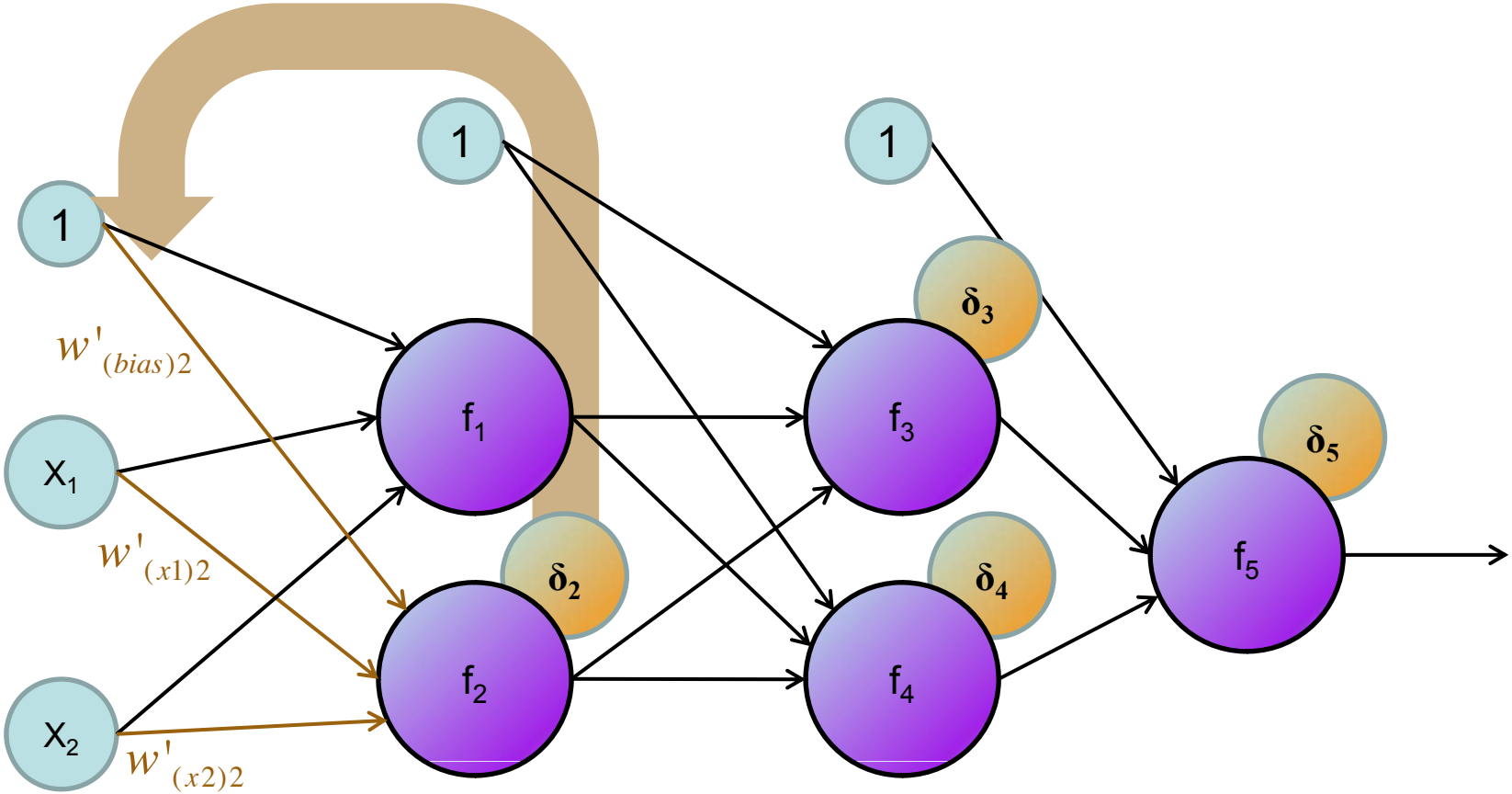


$$w'_{(bias)1} = w_{(bias)1} - \eta \cdot \delta_1$$

$$w'_{(x1)1} = w_{(x1)1} - \eta \cdot \delta_1 \cdot x_1$$

$$w'_{(x2)1} = w_{(x2)1} - \eta \cdot \delta_1 \cdot x_2$$

Backward Pass - Stage 2

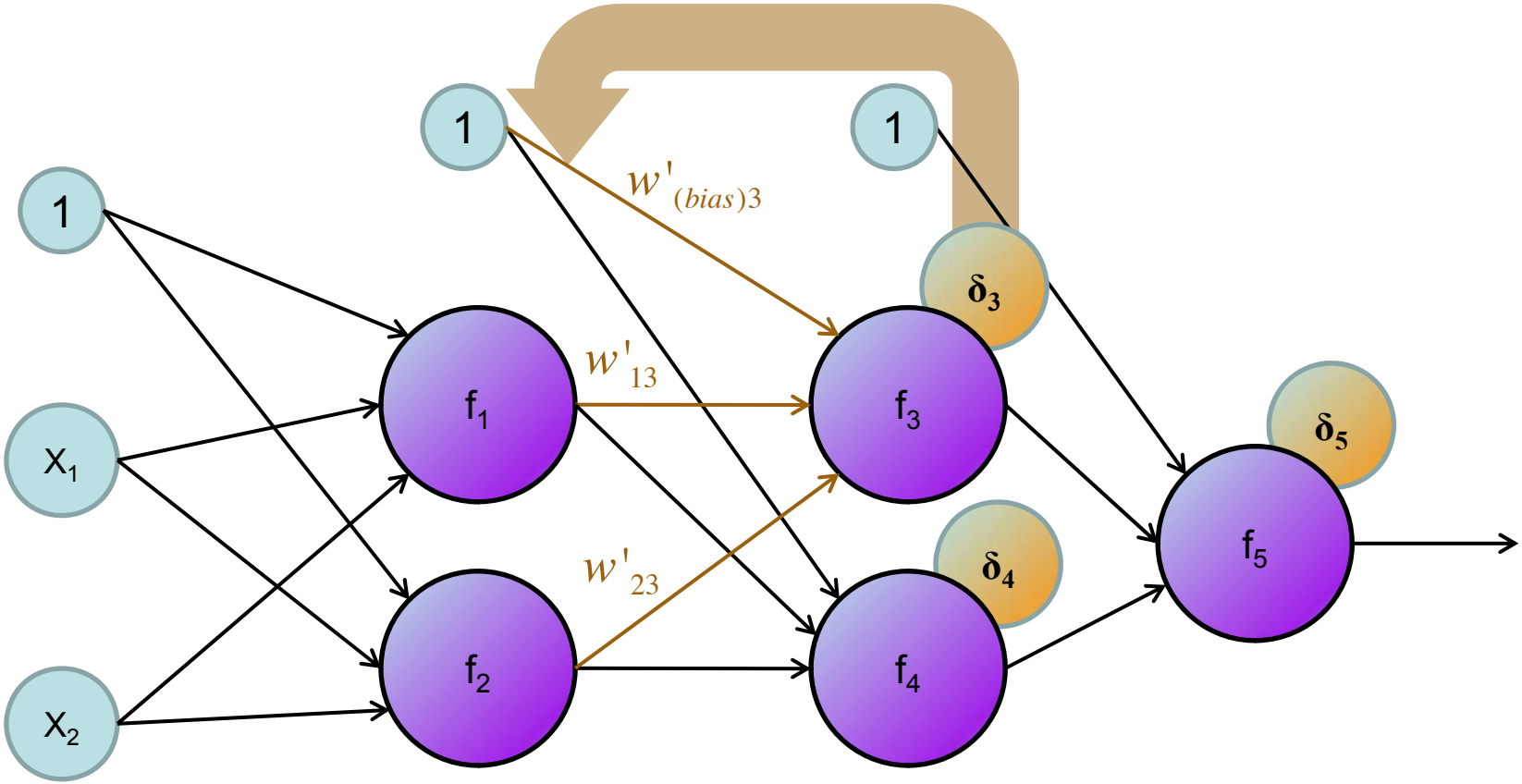


$$w'_{(bias)2} = w_{(bias)2} - \eta \cdot \delta_2$$

$$w'_{(x1)2} = w_{(x1)2} - \eta \cdot \delta_2 \cdot x_1$$

$$w'_{(x2)2} = w_{(x2)2} - \eta \cdot \delta_2 \cdot x_2$$

Backward Pass - Stage 2

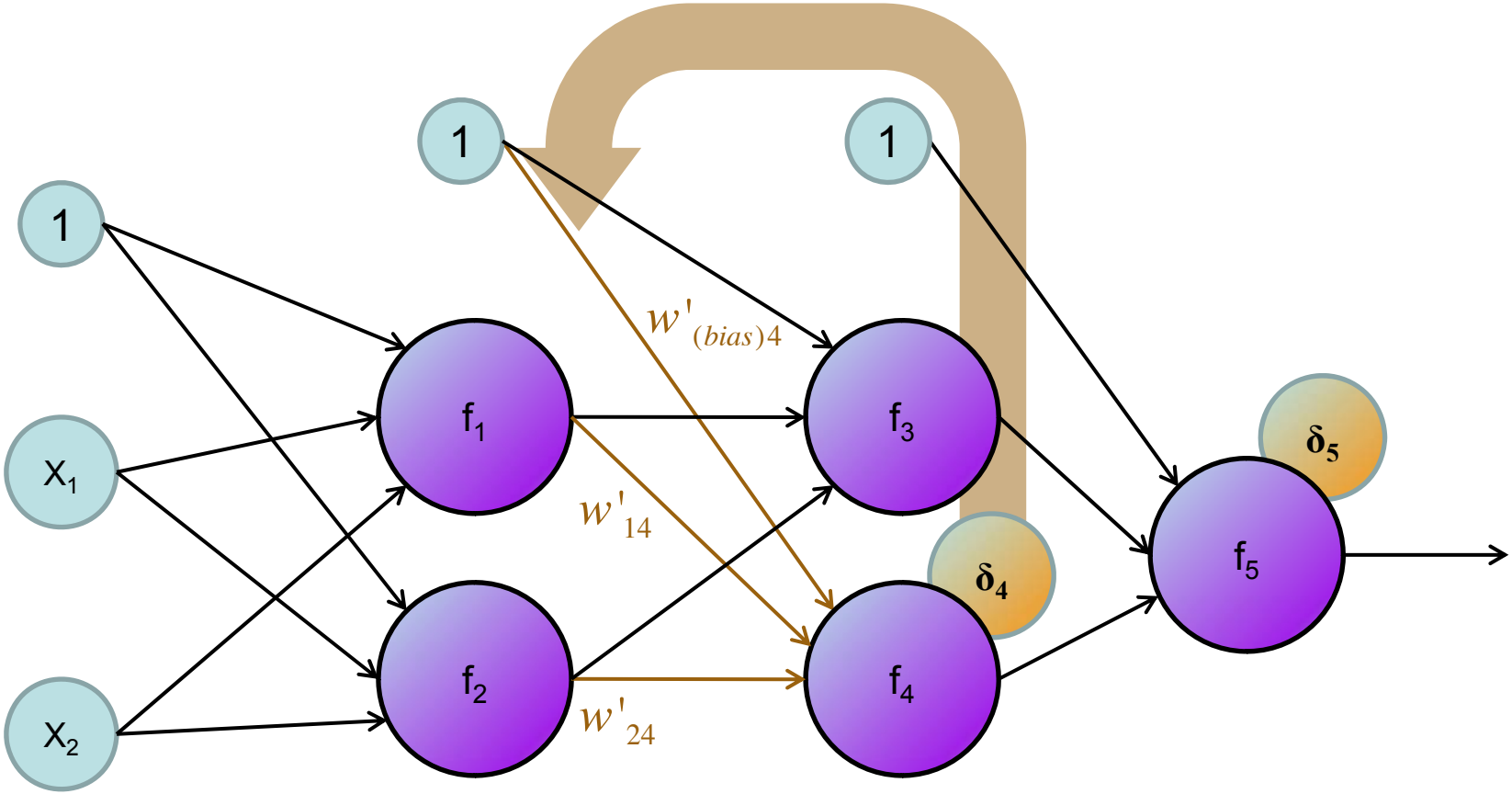


$$w'_{(bias)3} = w_{(bias)3} - \eta \cdot \delta_3$$

$$w'_{13} = w_{13} - \eta \cdot \delta_3 \cdot y_1$$

$$w'_{23} = w_{23} - \eta \cdot \delta_3 \cdot y_2$$

Backward Pass - Stage 2

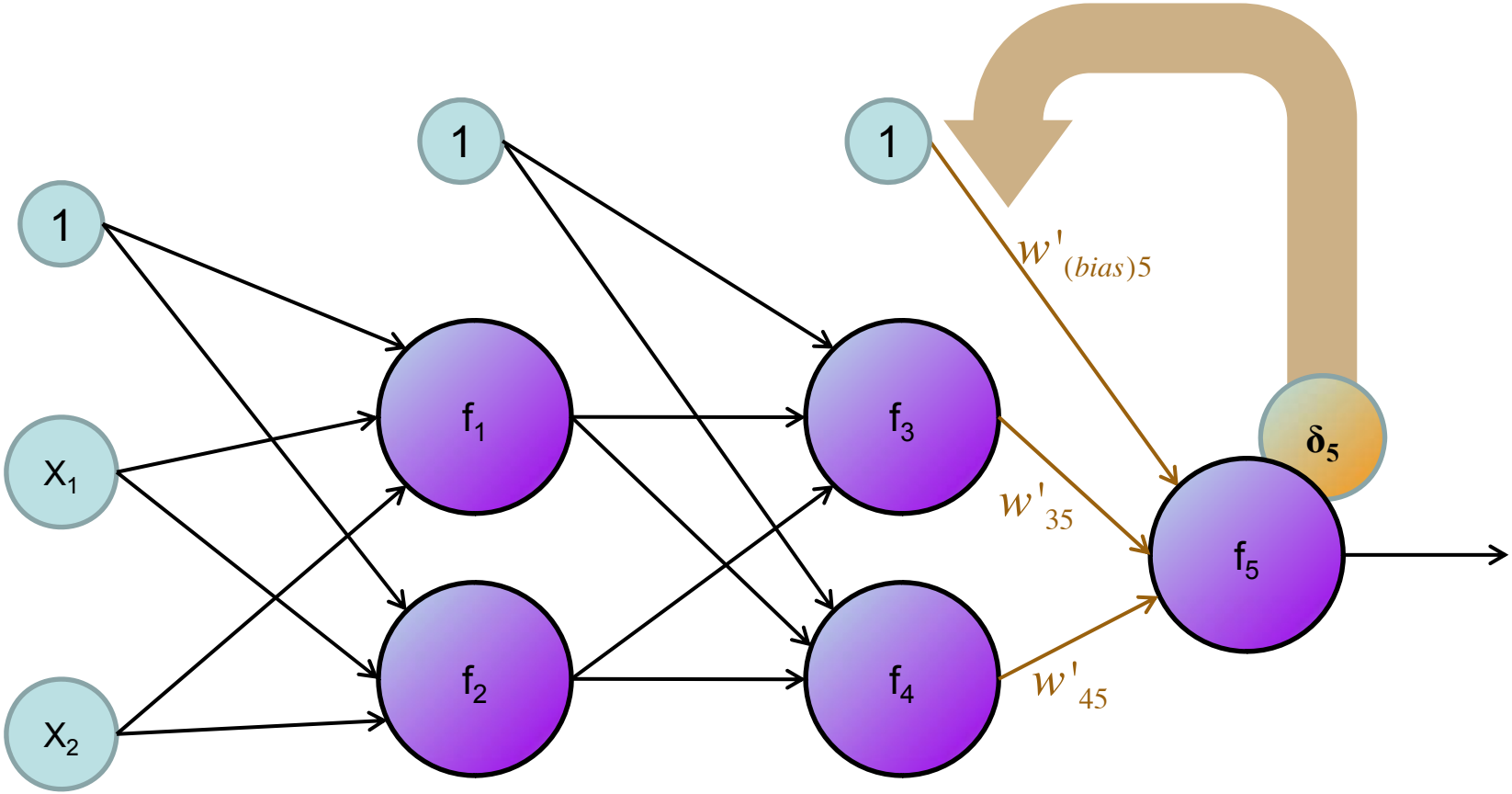


$$w'_{(bias)4} = w_{(bias)4} - \eta \cdot \delta_4$$

$$w'_{14} = w_{14} - \eta \cdot \delta_4 \cdot y_1$$

$$w'_{24} = w_{24} - \eta \cdot \delta_4 \cdot y_2$$

Backward Pass - Stage 2

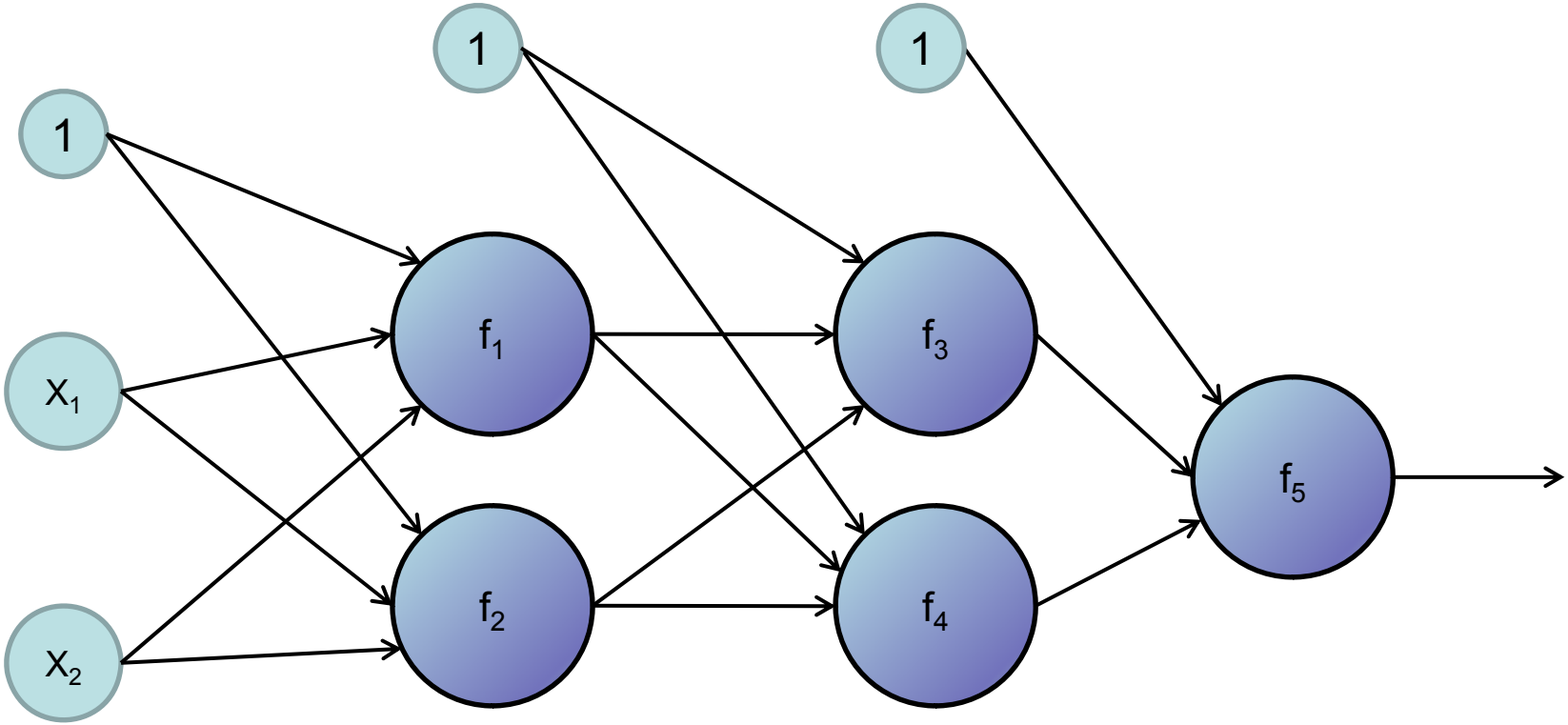


$$w'_{(bias)5} = w_{(bias)5} - \eta \cdot \delta_5$$

$$w'_{35} = w_{35} - \eta \cdot \delta_5 \cdot y_3$$

$$w'_{45} = w_{45} - \eta \cdot \delta_5 \cdot y_4$$

Pattern p+1 is presented



The procedure is the same as before…

Notes

- In backward pass you could update the weights immediately after you calculate the deltas.

- Here the backward pass was done in 2 distinct stages: a) propagation of errors backwards in order to evaluate the derivatives and b) weight adjustment using the calculated derivatives.

- In class notes the derivative of the error is (target – output). If you prefer it like this then you need to put a plus sign (+) in front of the learning rate in the weight update equations (as in class notes).

- A large learning rate is equivalent to big changes in the weights, thus large jumps in the weight space. This is not always desirable.

Notes

- To minimise the occurences of local minima:
  - Change the learning rate (either start with a large value and progressively decrease it or intelligently adapt it)
  - Add more hidden nodes (be careful of the overfitting problem)
  - Add momentum to the weight update equations
  - Add noise
- Overfitting occurs when the neural network:
  - is trained for too long (to avoid this problem stop training early)
  - has a lot of hidden nodes (to avoid this problem do model selection)
- In the batch update version the weight changes are accumulated and applied after each epoch instead of after each pattern.
- Weights should be initialised to small random values in the range [-1,1] and input data should be normalised in the range [0,1].