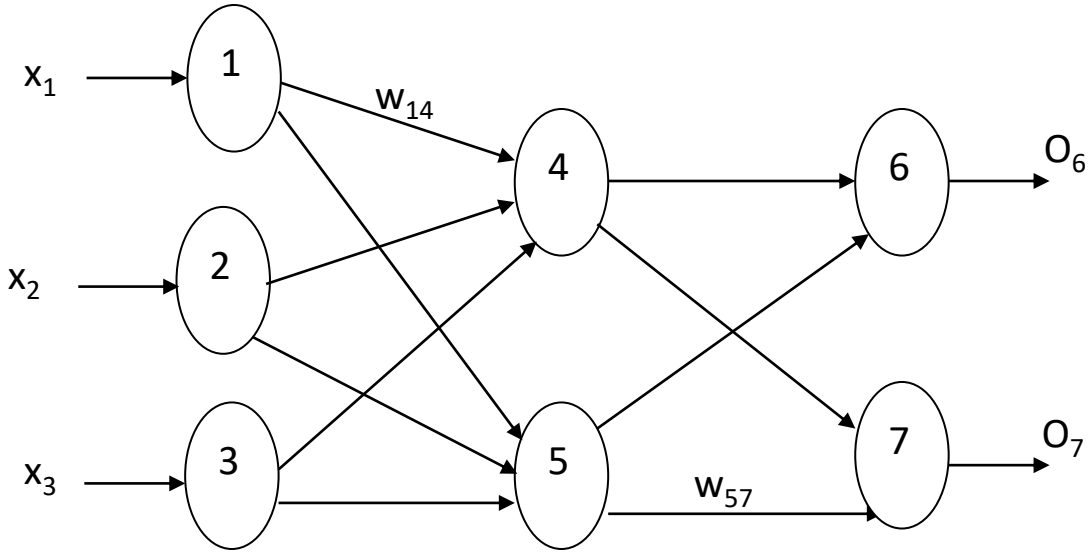
D) Example of computations of NNL by Backpropagation algorithm



Input:

$x_1 = 1$

$x_2 = 0$

$x_3 = 1$

Weights:

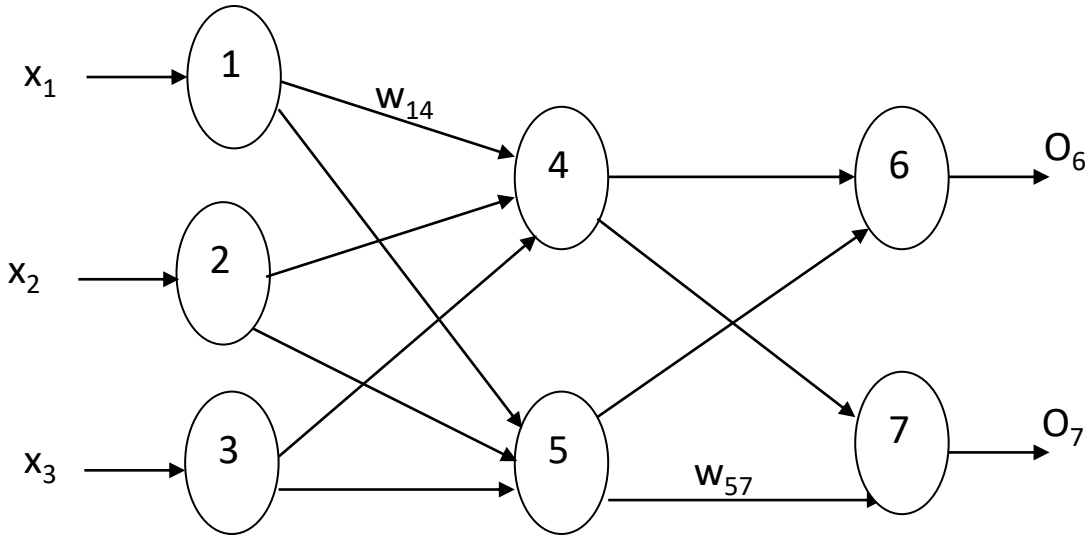
w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{47}	w_{56}	w_{57}
0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	0.1	-0.2	0.1

Biases:

θ_4	θ_5	θ_6	θ_7
-0.4	0.2	0.1	-0.1

i) Calculation of net input and output

Unit, j	I_j	O_j
4	(1*0.2)+(0*0.4)+(1*(-0.5))+(-0.4) = -0.7	$1/(1+e^{0.7}) = 0.332$
5	…	0.525
6 … 0.474;	7 … 0.496	



Input:

$x_1 = 1$

$x_2 = 0$

$x_3 = 1$

Biases:

θ_4	θ_5	θ_6	θ_7
-0.4	0.2	0.1	-0.1

Weights:

w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{47}	w_{56}	w_{57}
0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	0.1	-0.2	0.1

ii) Error calculation

- ✓ Output layer, $Err_j = O_j (1-O_j)(T_j - O_j)$ $[O_j(1-O_j) = df/dI_j$ – gradient descent]
- ✓ Hidden layer, $Err_j = O_j (1-O_j) \Sigma_k Err_k * w_{jk}$ [k – next layer unit]

Unit, j	Error	
6	0.474(1-0.474)(1-0.474)	≈ 0.1311
7	...	≈ 0.126
4	0.332 (1-0.332) [0.1311 * (-0.3) + 0.126*0.1]	≈ -0.0059
5	...	≈ -0.0034

True output:

$T_6 = 1$

$T_7 = 1$

Weights:

w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{47}	w_{56}	w_{57}
0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	0.1	-0.2	0.1

Biases:

θ_4	θ_5	θ_6	θ_7
-0.4	0.2	0.1	-0.1

iii) Updating Weights and Biases

- Weights: $w_{ij} = w_{ij} + \Delta w_{ij}$
  $\Delta w_{ij} = l * Err_j * O_i$
- Biases: $\theta_j = \theta_j + \Delta\theta_j$
  $\Delta\theta_j = l * Err_j$

Learning rate: Taken, l = 0.9; [0,1];
Helps avoid local maxima;
Heuristics; May be taken, l = 1/No. of iterations.

$w_{46} = -0.3 + 0.9 * 0.1311 * 0.332 \approx -0.261$

w_{47} = ……

w_{56}, w_{57}, w_{14}, w_{15}, w_{24}, w_{25}, w_{34}, w_{35}

$\theta_6 = 0.1 + 0.9 * 0.1311 \approx 0.218$

θ_7=……

θ_4, θ_5

E. Features of the Backpropagation Algorithm

1. Learning is arranged by iteratively processing training samples.
2. Weights and biases are updated based on 'mean squared error' between the network prediction (output) and the actual class (set label), propagated backward from the output layer toward the input layer, layer by layer.
3. In case of 'Case updating', which is usual, updates are done after each sample input is processed.
4. In case of 'Epoch updating' accumulated error after passing all samples of the training set is propagated backward.
5. Initial weights and biases are taken randomly from a range, say, -1.0 to 1.0 or -0.5 to 0.5.
6. The process terminates if, (a) The change after an epoch is less than threshold; or, (b) A fixed number of epochs have been run; or, (c) The performance (rate of misclassification) with respect to previous epoch is below a threshold.
7. Convergence is not guaranteed, 100s of 1000s of epochs may be required, and the frustration very often leads to setting everything anew, from training samples and network topology to weights and biases.