Backpropagation Algorithm with Gradient Descent

1. Forwards pass to compute a_n, z_n, b_n, and $\hat{y}_n$

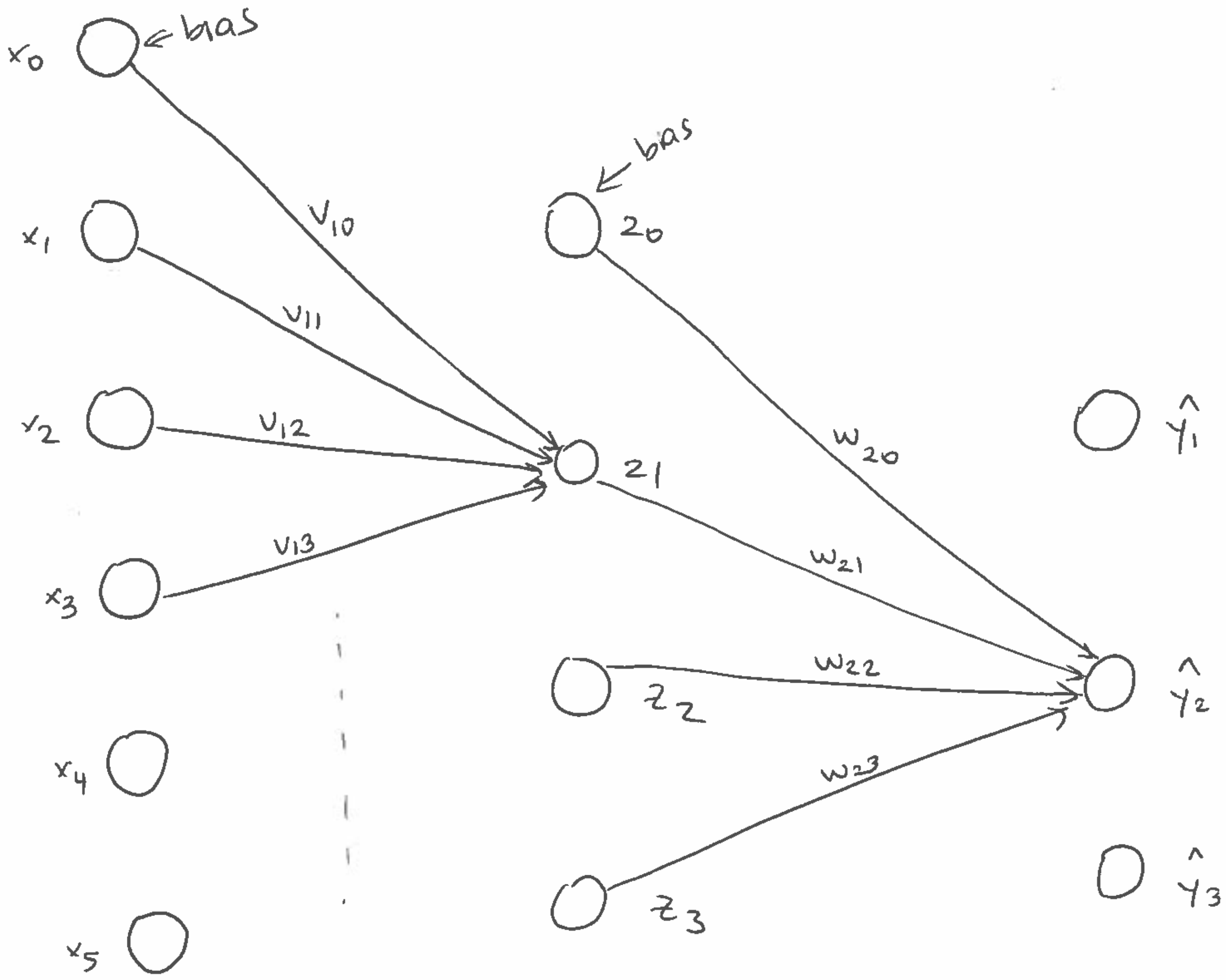
2. Compute the error signals for the output layer (δ_n^w)

3. Pass δ_n^w backwards to compute the error signals for the hidden layer (δ_n^v)

4. Compute the gradients (ΔW, ΔV)

5. Update the weights (W, V) (i.e, Gradient Descent)

Example Neural Network (for 3-class classification)



Not all connections are shown, but the network is fully
connected.

1. Forwards pass to compute $a_n, z_n, b_n, \hat{y}_n$

x_n : input vector for the n^{th} data instance (includes the bias node)

V : matrix of the weights from the input layer to the hidden layer

a_n : pre-synaptic hidden layer

z_n : post-synaptic hidden layer

W : matrix of the weights from the hidden layer to the output layer

b_n : pre-synaptic output layer

$\hat{y}_n$: post-synaptic output layer

Computations

$$a_n = V \cdot x_n$$

$$V \cdot x_n = \begin{bmatrix} v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{20} & v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \\ v_{30} & v_{31} & v_{32} & v_{33} & v_{34} & v_{35} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \rightarrow H \times 1 \text{ vector}$$

$H \times (D+1)$ $\qquad$ $(D+1) \times 1$

H: # of hidden units (excep bias)

$D+1$: # of input units (w/ bias)

$z_n = \text{sigm}(a_n)$ (thus z_n does not include the bias)

Add '1' as z_{n0} to z_n to represent the bias term.

$b_n = W \cdot z_n$

$$W \cdot z_n = \underbrace{\begin{bmatrix} w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ w_{30} & w_{31} & w_{32} & w_{33} \end{bmatrix}}_{K \times (H+1)} \underbrace{\begin{bmatrix} z_{n0} \\ z_{n1} \\ z_{n2} \\ z_{n3} \end{bmatrix}}_{(H+1) \times 1} \rightarrow K \times 1 \text{ vector}$$

K → # of output units

$(H+1)$ → # of hidden units (w/ bias)

$$\hat{y}_n = \text{softmax}(b_n)$$

$$\hat{y}_{n1} = \text{softmax}(b_n)_1 = \frac{e^{b_{n1}}}{\sum_{k=1}^{K} e^{b_{nk}}}$$

Example: If $K=3$ and $b_{n1} = 2.3$, $b_{n2} = 1.4$, $b_{n3} = 0.6$

$$\hat{y}_{n1} = \text{softmax}(b_n)_1 = \frac{e^{2.3}}{e^{2.3} + e^{1.4} + e^{0.6}}$$

$$\hat{y}_{n1} + \hat{y}_{n2} + \hat{y}_{n3} = 1$$

$$\hat{y}_{n2} = \text{softmax}(b_n)_2 = \frac{e^{1.4}}{e^{2.3} + e^{1.4} + e^{0.6}}$$

$$\hat{y}_{n3} = \text{softmax}(b_n)_3 = \frac{e^{0.6}}{e^{2.3} + e^{1.4} + e^{0.6}}$$

2. Compute the error signals for the output layer (δ_n^w)

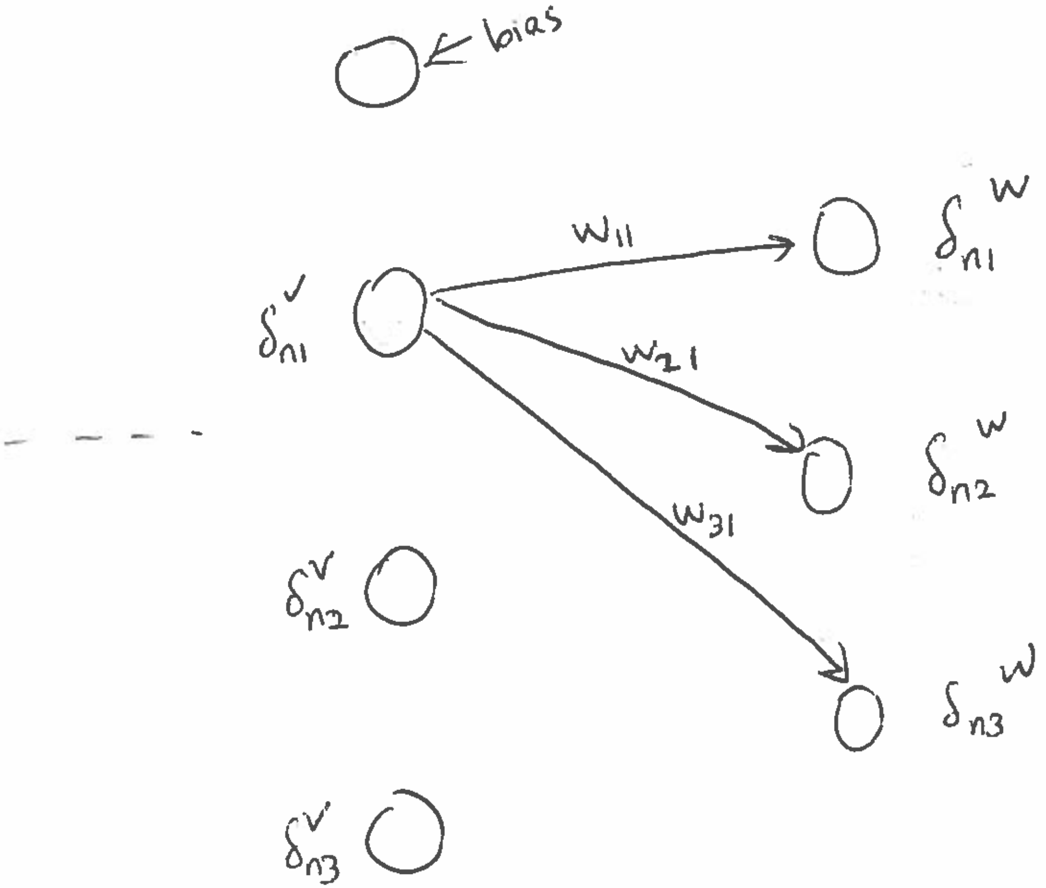
$$\delta_n^w = \hat{y}_n - y_n$$

vector for the error signals

E.g

$\hat{y}_n$	y_n	$\hat{y}_n - y_n$
0.6	1	-0.4
0.3	0	0.3
0.1	0	0.1

3. Pass δ_n^w backwards to compute the error signals for the hidden layer (δ_n^v)



Assume, the network is fully connected.

$$\delta_{n1}^v = (\delta_{n1}^w \cdot w_{11} + \delta_{n2}^w \cdot w_{21} + \delta_{n3}^w \cdot w_{31}) \cdot g'(a_{n1})$$

derivative of

pre-synaptic value at hidden

If g is the sigmoid function:

$$\text{sigm}'(a_{n1}) = \text{sigm}(a_{n1}) \cdot (1 - \text{sigm}(a_{n1}))$$

$$\text{sigm}(a_{n1}) = z_{n1}$$

Hence,

$$\delta_{n1}^{v} = (\delta_{n1}^{w} \cdot w_{11} + \delta_{n2}^{w} \cdot w_{21} + \delta_{n3}^{w} \cdot w_{31}) \cdot z_{n1} \cdot (1 - z_{n1})$$

Compute all δ_{nj}^{v} s in the same manner

4. Compute the gradients $(\Delta W, \Delta V)$

For each data instance n

$$\Delta w_{kj} = \delta^{w}_{nk} \cdot z_{nj}$$

$$\Delta v_{ji} = \delta^{v}_{nj} \cdot x_{ni}$$

For the entire dataset

$$\Delta w_{kj} = \sum_{n=1}^{N} \delta^{w}_{nk} \cdot z_{nj} \rightarrow \underbrace{\begin{bmatrix} \Delta w_{10} & \Delta w_{11} & \Delta w_{12} & \cdots \\ & & & \\ & & & \end{bmatrix}}_{\Delta W}$$

$$\Delta v_{ji} = \sum_{n=1}^{N} \delta^{v}_{nj} \cdot x_{ni}$$

5. Update the weights (W, V)

$$W = W - \eta \cdot \frac{1}{N} \Delta W$$

$$V = V - \eta \cdot \frac{1}{N} \Delta V$$

η : learning rate