HOW TO STAY AWAKE IN CLASS

— a guide —

- **HOT DRINK**
  rating: 4/10

COFFEE IS GREAT AND ALL, BUT IF YOU'RE REALLY TIRED THERE'S NO HOPE OF IT KEEPING YOU UP. ...UNLESS IT'S REALLY SCALDING.

- **HOT DRINK/COLD DRINK**
  rating: 6/10

ALTERNATE BETWEEN COFFEE/TEA AND SOMETHING COLD, LIKE A SLUSHIE. THE TEMPERATURE DIFFERENCE AND NAUSEATING TASTE COMBINATION CONSPIRE TO KEEP YOU CONSCIOUS.

- **HAVE NEIGHBOR PUNCH YOU**
  rating: 4.5/10

THEY'RE GENERALLY TOO NICE TO ACTUALLY DO IT. YOUR BEST BET IS TO TELL THEM THAT YOU ARE POSSIBLY CONCUSSED AND MIGHT NEVER WAKE UP IF ALLOWED TO SLEEP.

- **STAB SELF IN HAND WITH PEN**
  rating: 1.5/10

TYPICALLY ONLY WAKES YOU UP FOR A SECOND, ALTHOUGH THIS ONE'S EFFICACY REALLY DEPENDS ON YOUR WILLPOWER.

- **HOLD FEET OFF THE GROUND**
  rating: 2/10

YOU ALWAYS PUT YOUR FEET DOWN. AND THEN THERE'S NO HOPE.

- **EAT**
  rating: 9/10

ACTUALLY INCREDIBLY EFFECTIVE. JUST BE SURE NOT TO RUN OUT OF FOOD OR

- **GET ENOUGH SLEEP THE NIGHT BEFORE**
  rating: N/A

THIS WILL NEVER HAPPEN AS LONG AS THE INTERNET EXISTS.

ECS171: Machine Learning

L8 NN Backpropagation

Instructor: Prof. Maike Sonnewald
TAs: Pu Sun & Devashree Kataria

MOCO Amsterdam garden

Intended learning outcomes

- Explain what happens when weights are adjusted and how the different magnitudes impact the training efficiency and gradient descent
- Work through how we compute the output layer error with the gradient of the Loss Function
- Appreciate reasoning and how to apply termination criteria
- Describe the vanishing gradient problem

Training Neural Networks: Back-propagation

- The core algorithm for how neural networks learn
- Algorithm for how a single training example would like to nudge the weights and biases

Training neural nets:

Loop until convergence:

- for each example n
  1. Given input $\mathbf{x}^{(n)}$, propagate activity forward $(\mathbf{x}^{(n)} \rightarrow \mathbf{h}^{(n)} \rightarrow o^{(n)})$ (**forward pass**)
  2. Propagate gradients backward (**backward pass**)
  3. Update each weight (via gradient descent)

X is input, h is hidden layer and o is output

The benefit of backpropagation

Definition:

- An algorithm used for **efficiently** computing the gradients of the cost function with respect to each parameter
- It applies the chain rule (calculus) in a structured way moving backwards through the network
- Algorithm shows how a single training example would like to nudge the weights and biases

Purpose:

- Compute gradients in a computationally efficient manner.
- Without backpropagation, calculating the gradients, especially in large networks, would be extremely computationally expensive.

Hebbian learning in ML

- **Hebbian theory** is a neuropsychological theory claiming that an increase in synaptic efficacy arises from a presynaptic cell's repeated and persistent stimulation of a postsynaptic cell (wikipedia)
- “Neurons that fire together wire together”
  - Strengthening of connections happens between neurons that are the most active and connected
- Neural network with at least one hidden layer is a universal approximator (can represent any function)
  - Proof in: Approximation by Superpositions of Sigmoidal Function, Cybenko, paper
- The capacity of the network increases with more hidden units and more hidden layers

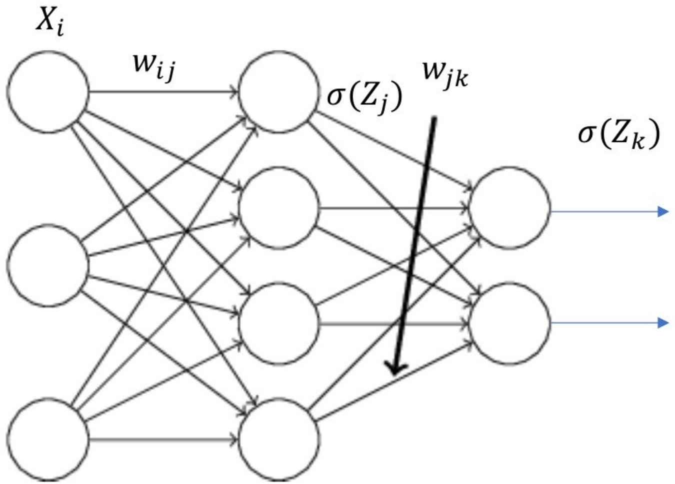
Donald Hebb (1904-1985) Can.

Key Idea behind Backpropagation

- We don’t have targets for a hidden unit, but we can compute how fast the error changes as we change its activity
- Instead of using desired activities to train the hidden units, use error derivatives w.r.t. hidden activities
- Each hidden activity can affect many output units and can therefore have many separate effects on the error
    - These effects must be combined
- We can compute error derivatives for all the hidden units efficiently
- Once we have the error derivatives for the hidden activities, it’s easy to get the error derivatives for the weights going into a hidden unit: This is just the chain rule!

Neural Network

Sigmoid (x) : $\sigma(x)$ = 1 / (1 + e^{-x})



i: related to input layer
j : related to hidden layer
k: related to output layer
K: number of neurons in the output layer

b_j : biases in the hidden layer.
b_k : biases in the output layer.

X_i: input information in the input layer.
W_{ij}: weights connecting input to hidden layer.
W_{jk}: weights connecting hidden layer to output layer.

$$Z_j = W_{ij}X_i + b_j = \sum_i w_{ij}x_i + b_j$$
$$Z_k = W_{jk}\,\sigma(Z_j) + b_k$$
$$\hat{y}_k = \sigma(Z_k)$$
$$predicted\ output\text{:}\ \hat{y}_k = \sigma(W_{jk}\,\sigma(W_{ij}X_i + b_j) + b_k)$$
$$actual\ output:\ y$$

Neural network error function E(w)

N= number of samples

$$E(w) = \sum_{n=1}^{N} E_n(w)$$

E_n: Error evaluation for the n^{th} observation

$$E_n = \frac{1}{2}\sum_{k}(\hat{y}_{nk} - y_n)^2$$

k: number of output nodes

Sum of Squared Errors for all dataset observations

$$E(w) = \sum_{records}\ \sum_{output\ nodes} (predicted - actual)^2$$

b_j : biases in the hidden layer.

b_k : biases in the output layer.

X_i: data coming from the input layer.

W_{ij}: weights connecting input to hidden layer.

W_{jk}: weights connecting hidden layer to output layer.

$$Z_j = W_{ij}X_i + b_j = \sum_i w_{ij}x_i + b_j$$

$$X_j = \sigma\left(W_{ij}X_i + b_j\right) \quad \text{: data coming from the hidden layer}$$

$$Z_k = W_{jk}\,\sigma Z_j + b_k$$

$$\hat{y}_k = \sigma(Z_k)$$

$$\hat{y}_k = \sigma(W_{jk}\,\sigma(W_{ij}X_i + b_j) + b_k)$$

Compute the output layer error with the gradient of the Loss Function

b_j : hidden layer biases
b_k : output layer biases

X_i: data in the input layer.
W_{ij}: hidden layer weights.
W_{jk}: output layer weights.

The output layer error:

$$\frac{\partial E_n(w)}{\partial W_{jk}} = \frac{\partial\left(\frac{1}{2}\sum_k(\hat{y}_{nk} - y_n)^2\right)}{\partial W_{jk}}$$

$$= (\hat{y}_{nk} - y_{nk})\frac{\partial\,(\hat{y}_{nk} - y_n)}{\partial W_{jk}} = (\hat{y}_{nk} - y_n)\frac{\partial\,(\hat{y}_{nk})}{\partial W_{jk}}$$

We know that the predicted outcome for data point n in out output layer k is:

$$\hat{y}_{nk} = \sigma(W_{jk}\,\sigma(W_{ij}X_i + b_j) + b_k) \quad \text{or} \quad \hat{y}_{nk} = \sigma(Z_k)$$

We write $X_j = \sigma\left(W_{ij}X_i + b_j\right)$ and for convenience: $X_j = \sigma(Z_j)$

Using the chain rule: $\frac{\partial E_n(w)}{\partial W_{jk}} = (\hat{y}_{nk} - y_n)\frac{\partial\,(\hat{y}_{nk})}{\partial W_{jk}} = (\hat{y}_{nk} - y_n)\frac{\partial(\hat{y}_{nk})}{\partial Z_k}\frac{\partial Z_k}{\partial W_{jk}}$

Next ->

Compute the output layer error with the gradient of the Loss Function

b_j : hidden layer biases
b_k : output layer biases

X_i: data in the input layer.
W_{ij}: hidden layer weights.
W_{jk}: output layer weights.

We have:

$$\frac{\partial E_n(w)}{\partial W_{jk}} = (\hat{y}_{nk} - y_n)\frac{\partial(\hat{y}_{nk})}{\partial Z_k}\frac{\partial Z_k}{\partial W_{jk}}$$

The derivative of sigmoid function $\sigma(x)$: $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$

Where Sigmoid(x): $\sigma(x) = 1/(1 + e^{-x})$

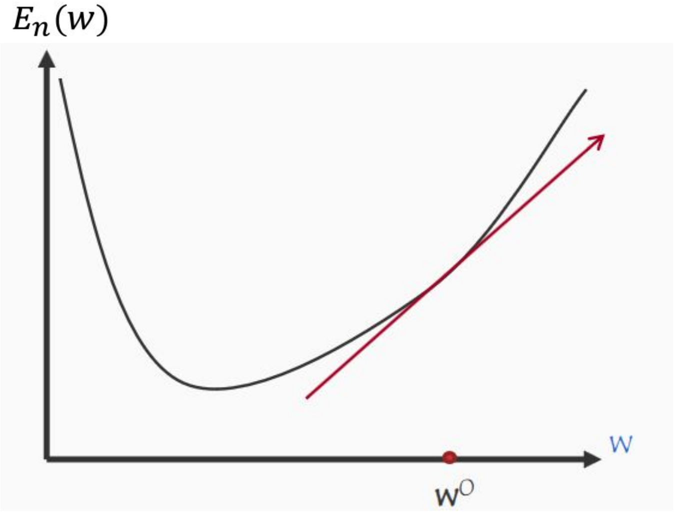
Thus:

$$\frac{\partial E_n(w)}{\partial W_{jk}} = (\hat{y}_{nk} - y_n)\,\sigma(Z_k)(1 - \sigma(Z_k))X_j$$

Backpropagation: Output to hidden layer

Using the gradient descent update rule: $\mathbf{w}_{new} \leftarrow \mathbf{w}_{current} + \Delta\mathbf{w}$

The $\Delta\mathbf{w}$ is now given by: $\frac{\partial E_n(w)}{\partial W_{jk}} = (\hat{y}_{nk} - y_n)\,\sigma(Z_k)(1-\sigma(Z_k))X_j$



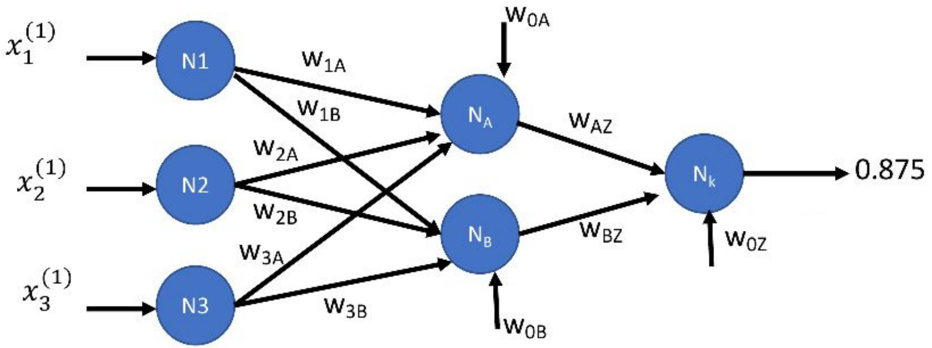
As such, varying the $\mathbf{w}_{current}$ value gets closer to the optimal weight

Useful derivatives for different activation functions

name	function	derivative
Sigmoid	$\sigma(z) = \frac{1}{1+\exp(-z)}$	$\sigma(z) \cdot (1 - \sigma(z))$
Tanh	$\tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$	$1/\cosh^2(z)$
ReLU	$\mathrm{ReLU}(z) = \max(0, z)$	$\begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z \leq 0 \end{cases}$

Backward pass: Updating W_{AZ}

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

$x_1 = N_1 = 0.4$	$N_A = 0.7892$
$x_2 = N_2 = 0.2$	$N_B = 0.8176$
$x_3 = N_3 = 0.7$	$N_z = 0.875$

$w_{0A} = 0.5$	$w_{0B} = 0.7$	$w_{0Z} = 0.5$
$w_{1A} = 0.6$	$w_{1B} = 0.9$	$w_{AZ} = 0.9$
$w_{2A} = 0.8$	$w_{2B} = 0.8$	$w_{BZ} = 0.9$
$w_{3A} = 0.6$	$w_{3B} = 0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

Backward pass: Updating W_{AZ}

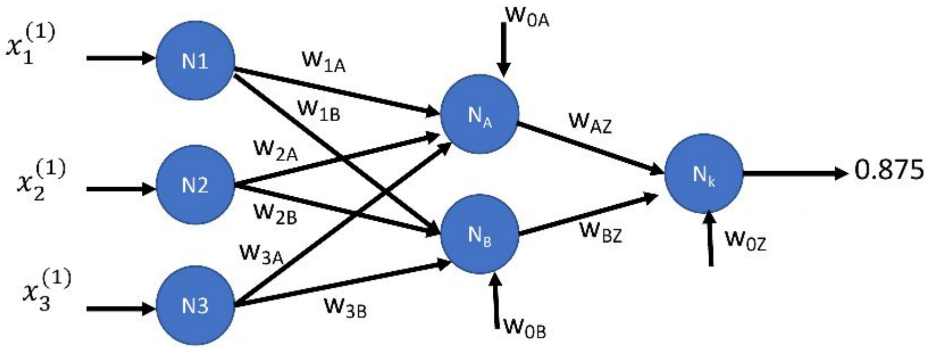
$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass

$x_1 = N_1 = 0.4$	$N_A = 0.7892$
$x_2 = N_2 = 0.2$	$N_B = 0.8176$
$x_3 = N_3 = 0.7$	$N_z = 0.875$

$w_{0A} = 0.5$	$w_{0B} = 0.7$	$w_{0Z} = 0.5$
$w_{1A} = 0.6$	$w_{1B} = 0.9$	$w_{AZ} = 0.9$
$w_{2A} = 0.8$	$w_{2B} = 0.8$	$w_{BZ} = 0.9$
$w_{3A} = 0.6$	$w_{3B} = 0.4$	



Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

Backward pass: Updating W_{AZ}

$learning\ rate\ ; 0 \le \eta \le 1$
$Assume:\ \eta = 0.1$

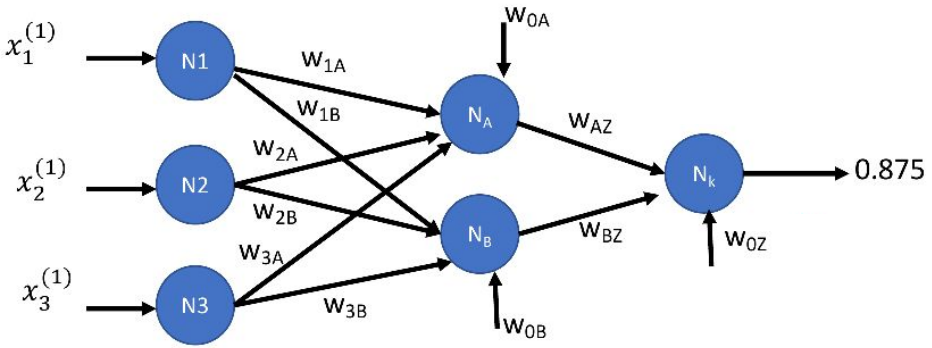
X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.9$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass

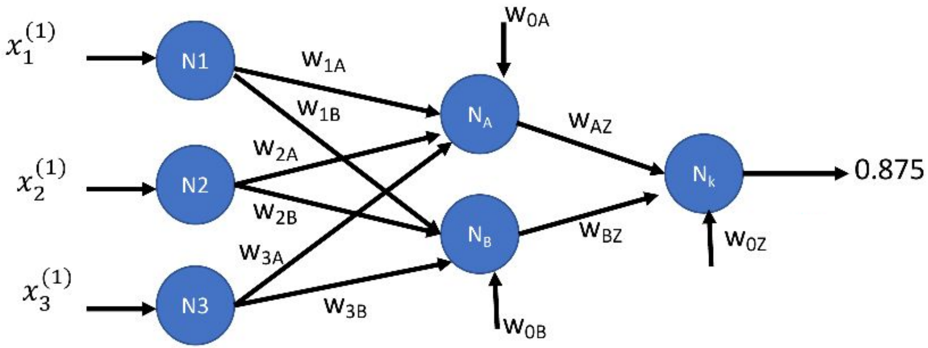


$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

Backward pass: Updating W_{AZ}

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

Z_k

$learning\ rate\ ; 0 \leq \eta \leq 1$

$Assume:\ \eta = 0.1$

X_i: N_1, N_2, N_3

$\sigma(Z_j)$: N_A, N_B

$\sigma(Z_k)$: N_z

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.9$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

Backward pass: Updating W_{AZ}

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass

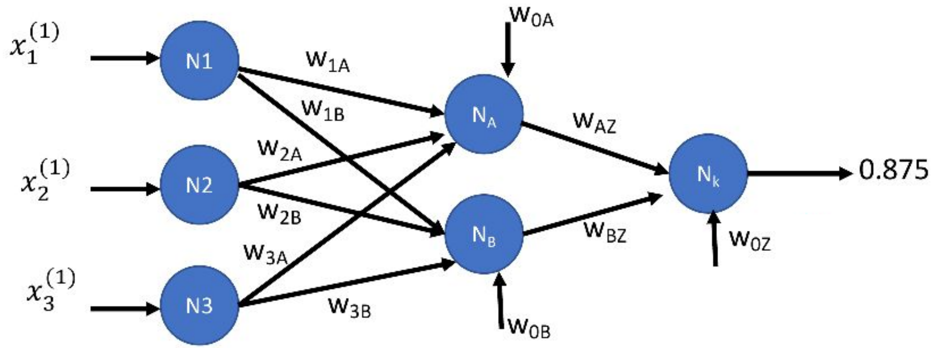
$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.9$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075



$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

Backward pass: Updating W_{AZ}

$learning\ rate\ ; 0 \leq \eta \leq 1$

$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3

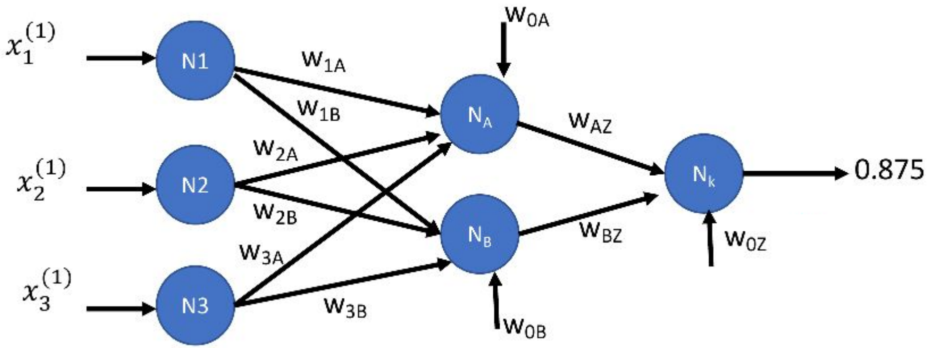
$\sigma(Z_j)$: N_A, N_B

$\sigma(Z_k)$: N_z

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.9$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	



Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta \Delta w_{current}$$

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

$$\Delta w_{current} = (\hat{y}_k - y)\,\sigma(Z_k)(1 - \sigma(Z_k))\sigma(Z_j)$$
$$= residual\ error * \sigma(Z_k)(1 - \sigma(Z_k))\sigma(Z_{N_A})$$

Backward pass: Updating W_{AZ}

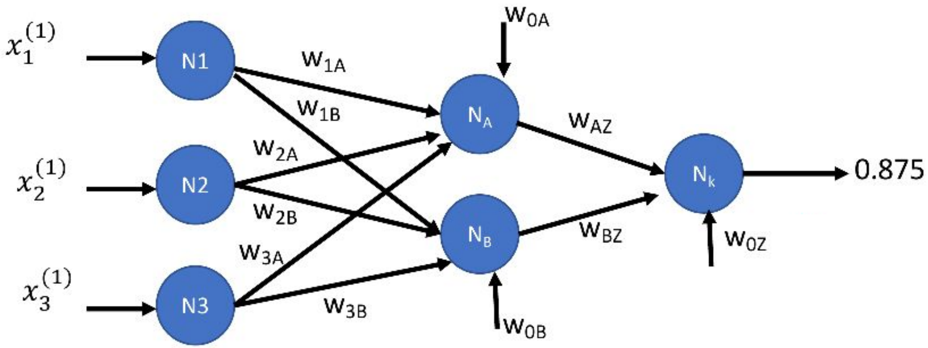
- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass

$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume:\ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.9$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	



Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta \Delta w_{current}$$

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

$$\Delta w_{current} = (\hat{y}_k - y)\,\sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_j)$$
$$= residual\ error * \sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_{N_A})$$

$$w_{AZ-new} = w_{AZ} + \eta \Delta w_{current}$$
$$= 0.9 + (0.1 \times 0.0067) = 0.90067$$

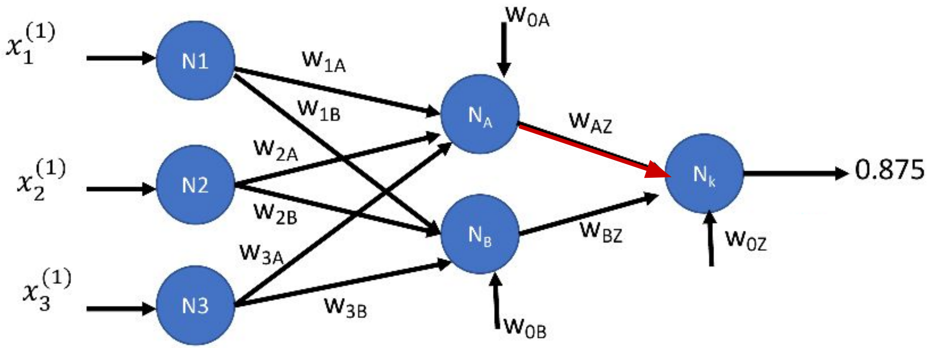
Backward pass: Updating W_{AZ}

learning rate; $0 \le \eta \le 1$
Assume: $\eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$



$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.90067$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta \Delta w_{current}$$

$$\Delta w_{current} = (\hat{y}_k - y)\,\sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_j)$$
$$= residual\ error * \sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_{N_A})$$

$$w_{AZ-new} = w_{AZ} + \eta \Delta w_{current}$$
$$= 0.9 + (0.1 \times 0.0067) = 0.90067$$

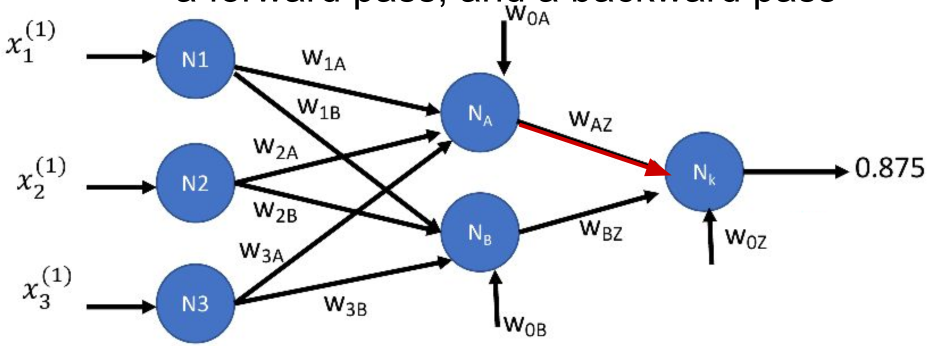
$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$
$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$
$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

Backward pass: Updating W_{BZ}

$learning\ rate\ ;0 \leq \eta \leq 1$
$Assume:\ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.90067$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

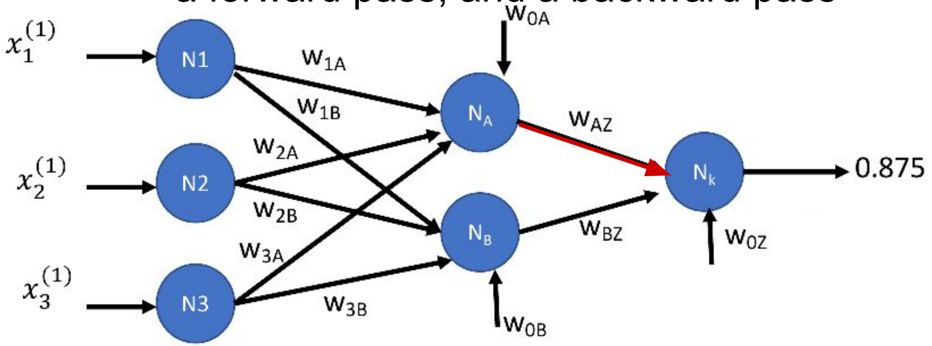
Backward pass: Updating W_{BZ}

$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.90067$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

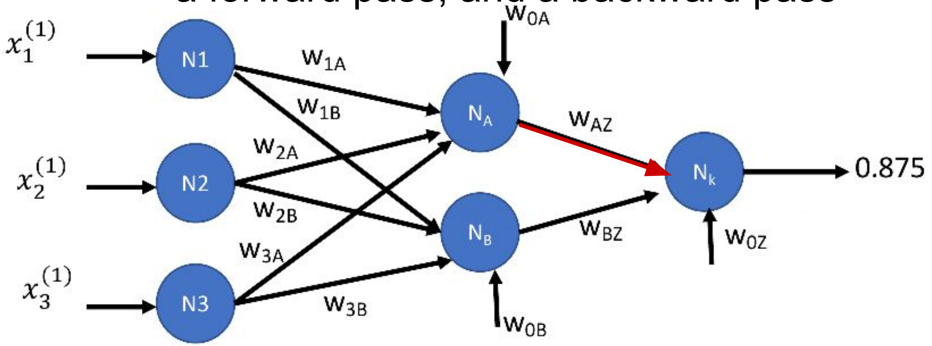
$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

Backward pass: Updating W_{BZ}

$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume:\ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



x_1=N_1 =0.4	N_A =0.7892
x_2=N_2 =0.2	N_B =0.8176
x_3=N_3 =0.7	N_z =0.875

w_{0A} =0.5	w_{0B} =0.7	w_{0Z}=0.5
w_{1A} = 0.6	w_{1B} = 0.9	w_{AZ}=0.90067
w_{2A}=0.8	w_{2B}=0.8	w_{BZ}=0.9
w_{3A}=0.6	w_{3B}=0.4	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta \Delta w_{current}$$

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A} x_1^{(1)} + \omega_{2A} x_2^{(1)} + \omega_{3A} x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B} x_1^{(1)} + \omega_{2B} x_2^{(1)} + \omega_{3B} x_3^{(1)} = 1.5$$

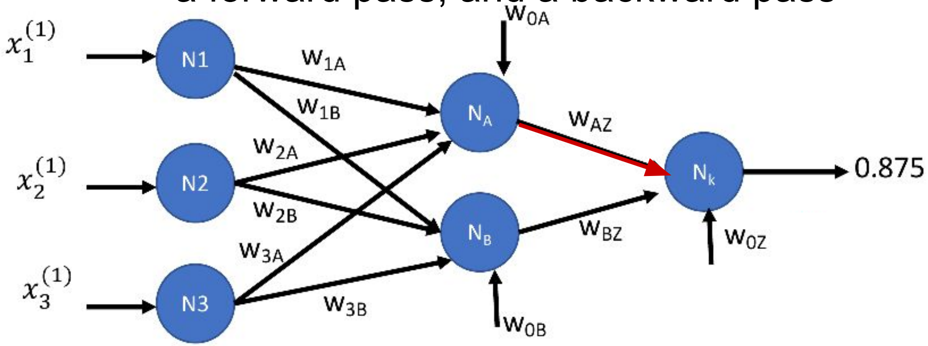
$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

Backward pass: Updating W_{BZ}

$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.90067$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta \Delta w_{current}$$

$$\Delta w_{current} = (\hat{y}_k - y)\,\sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_j)$$
$$= residual\ error * \sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_{N_B})$$

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

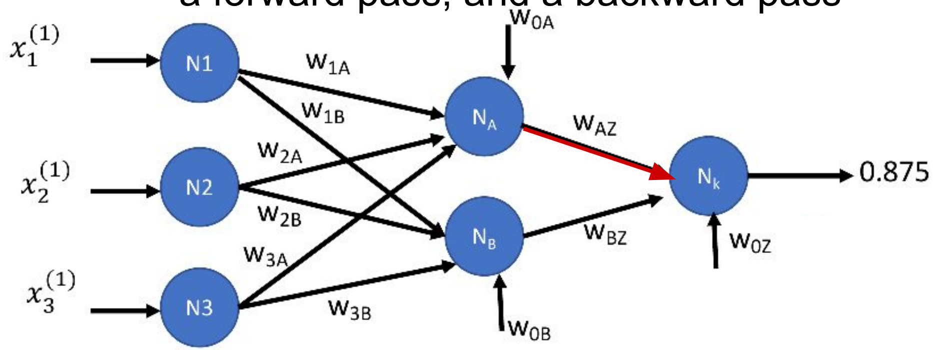
$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

Backward pass: Updating W_{BZ}

$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume:\ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.90067$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.9$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta \Delta w_{current}$$

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

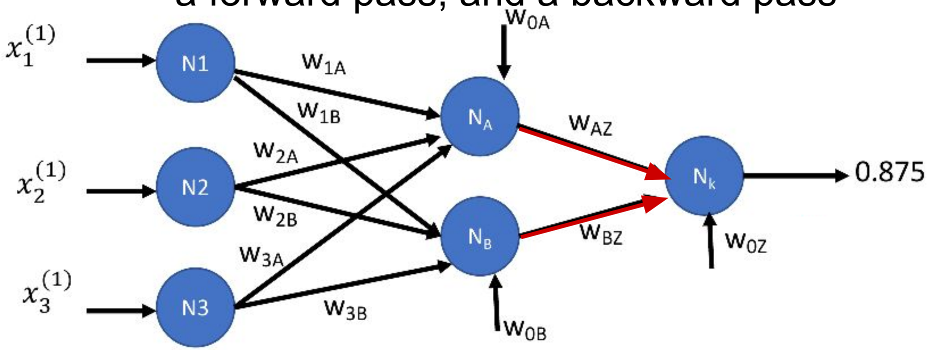
$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

$$\Delta w_{current} = (\hat{y}_k - y)\,\sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_j)$$
$$= residual\ error\ *\ \sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_{N_B})$$

$$w_{BZ-new} = w_{BZ} + \eta \Delta w_{current}$$
$$= 0.9 + (0.1 \times 0.0069) = 0.90069$$

Backward pass: Updating W_{BZ}

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



learning rate ; $0 \leq \eta \leq 1$

Assume : $\eta = 0.1$

X_i: N_1, N_2, N_3

$\sigma(Z_j)$: N_A, N_B

$\sigma(Z_k)$: N_z

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.90067$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.90069$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta \Delta w_{current}$$

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

$$\Delta w_{current} = (\hat{y}_k - y)\,\sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_j)$$
$$= residual\ error * \sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_{N_B})$$

$$w_{BZ-new} = w_{BZ} + \eta \Delta w_{current}$$
$$= 0.9 + (0.1 \times 0.0069) = 0.90069$$

Backward pass: Updating W_{0Z}

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass

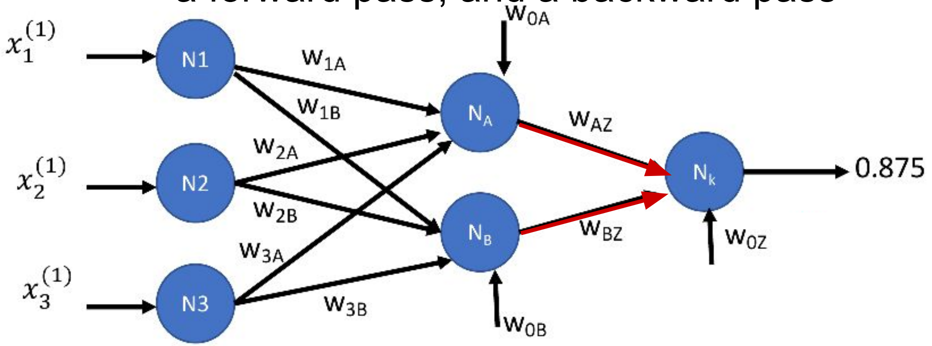
$learning\ rate\ ; 0 \leq \eta \leq 1$

$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3

$\sigma(Z_j)$: N_A, N_B

$\sigma(Z_k)$: N_z



$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.90067$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.90069$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

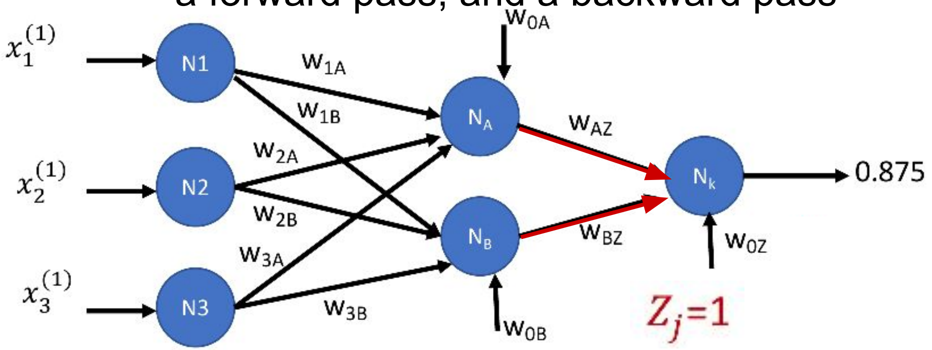
$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

Backward pass: Updating W_{0Z}

$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.90067$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.90069$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

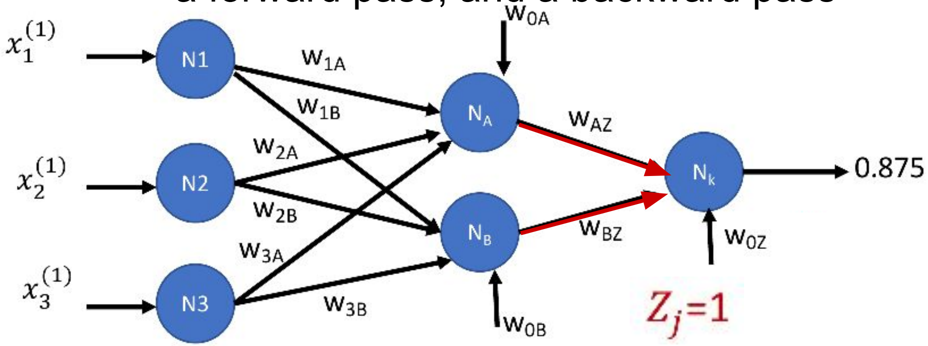
Backward pass: Updating W_{0Z}

$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume:\ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

$x_1 = N_1 = 0.4$	$N_A = 0.7892$
$x_2 = N_2 = 0.2$	$N_B = 0.8176$
$x_3 = N_3 = 0.7$	$N_z = 0.875$

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



$w_{0A} = 0.5$	$w_{0B} = 0.7$	$w_{0Z} = 0.5$
$w_{1A} = 0.6$	$w_{1B} = 0.9$	$w_{AZ} = 0.90067$
$w_{2A} = 0.8$	$w_{2B} = 0.8$	$w_{BZ} = 0.90069$
$w_{3A} = 0.6$	$w_{3B} = 0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta \Delta w_{current}$$

$$\Delta w_{current} = (\hat{y}_k - y)\,\sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_j)$$

$$= 0.075 * 0.87 * 0.13 * 1 = 0.008$$

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$

$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

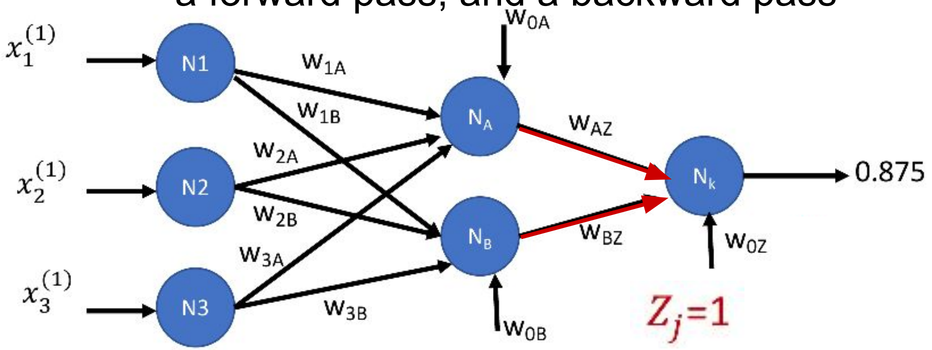
Backward pass: Updating W_{0Z}

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass

$learning\ rate;\ 0 \le \eta \le 1$
$Assume:\ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$



$w_{0A}=0.5$	$w_{0B}=0.7$	$w_{0Z}=0.5$
$w_{1A}=0.6$	$w_{1B}=0.9$	$w_{AZ}=0.90067$
$w_{2A}=0.8$	$w_{2B}=0.8$	$w_{BZ}=0.90069$
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta \Delta w_{current}$$

$$\Delta w_{current} = (\hat{y}_k - y)\,\sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_j)$$

$$= 0.075 * 0.87 * 0.13 * 1 = 0.008$$

$$w_{0Z-new} = w_{0Z} + \eta \Delta w_{current}$$

$$= 0.9 + (0.1 \times 0.0069) = 0.90069$$

$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

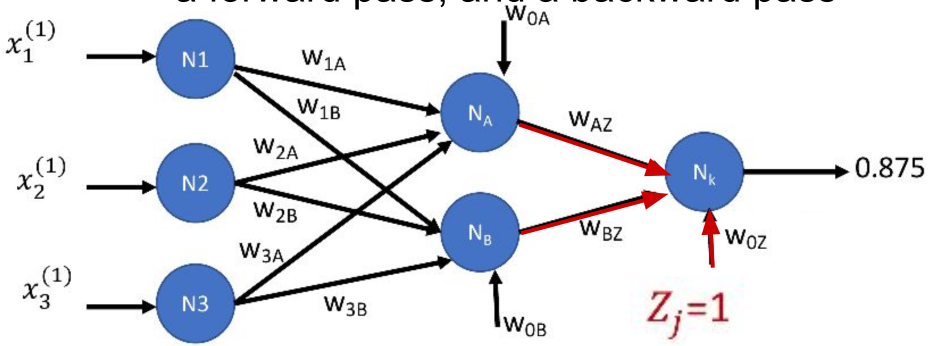
$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$

$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

Backward pass: Updating W_{0Z}

- Feed forward neural network learning in two phases:
  - a forward pass, and a backward pass



$$Z_j = Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

$$Z_{N_B} = \omega_{0B} + \omega_{1B}x_1^{(1)} + \omega_{2B}x_2^{(1)} + \omega_{3B}x_3^{(1)} = 1.5$$

$$Z_k = \omega_{0Z} + \omega_{AZ}\,\sigma(1.32) + \omega_{BZ}\,\sigma(1.5)$$
$$= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461$$

$learning\ rate\ ; 0 \leq \eta \leq 1$

$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3

$\sigma(Z_j)$: N_A, N_B

$\sigma(Z_k)$: N_z

$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	w_{0Z}=**0.5008**
$w_{1A}=0.6$	$w_{1B}=0.9$	w_{AZ}=0.90067
$w_{2A}=0.8$	$w_{2B}=0.8$	w_{BZ}=0.90069
$w_{3A}=0.6$	$w_{3B}=0.4$	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$w_{new} = w_{current} + \eta\Delta w_{current}$$

$$\Delta w_{current} = (\hat{y}_k - y)\,\sigma(Z_k)(1-\sigma(Z_k))\sigma(Z_j)$$
$$= 0.075 * 0.87 * 0.13 * 1 = 0.008$$

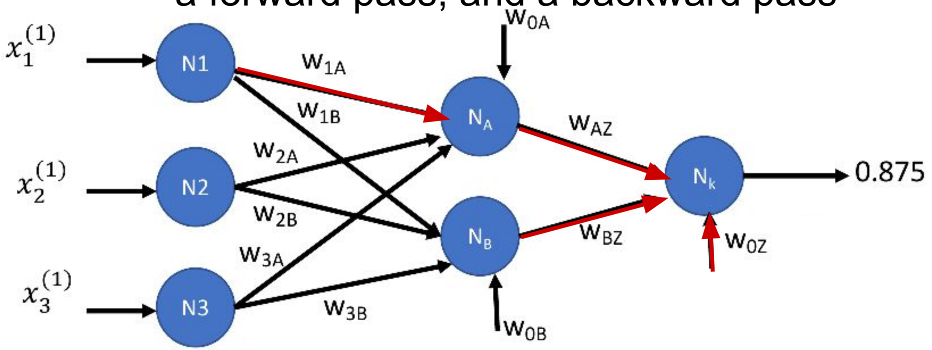
$$w_{0Z-new} = w_{0Z} + \eta\Delta w_{current}$$
$$= 0.5 + (0.1 \times 0.008) = 0.5008$$

Backward pass: Updating W_{1A}

$learning\ rate\ ; 0 \leq \eta \leq 1$
$Assume: \ \eta = 0.1$

X_i: N_1, N_2, N_3
$\sigma(Z_j)$: N_A, N_B
$\sigma(Z_k)$: N_z

- Feed forward neural network learning in two phases:
    - a forward pass, and a backward pass



$x_1=N_1=0.4$	$N_A=0.7892$
$x_2=N_2=0.2$	$N_B=0.8176$
$x_3=N_3=0.7$	$N_z=0.875$

$w_{0A}=0.5$	$w_{0B}=0.7$	w_{0Z}=**0.5008**
$w_{1A}=0.6$	$w_{1B}=0.9$	w_{AZ}=0.90067
w_{2A}=0.8	w_{2B}=0.8	w_{BZ}=0.90069
w_{3A}=0.6	w_{3B}=0.4	

Assume actual y= 0.8 →
residual error = 0.875 – 0.8 = 0.075

$$Z_{N_A} = \omega_{0A} + \omega_{1A}x_1^{(1)} + \omega_{2A}x_2^{(1)} + \omega_{3A}x_3^{(1)} = 1.32$$

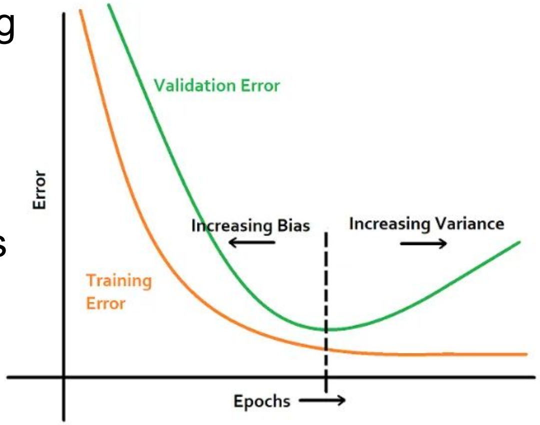
$$\Delta w_{hidden} = residual\ error * error\ of\ the\ hidden\ layer * weighted\ error\ of\ the\ output\ layer$$

$$\Delta w_{hidden} = (\hat{y} - y) * \frac{\partial \sigma(Z_j)}{\partial W_{ij}} * W_{jk} \frac{\partial \sigma(Z_k)}{\partial W_{jk}}$$

Termination Criteria: When to stop training

Epoch: number of times the learning algorithm works through the training data
Batch: Epochs have one or more batches

- Criteria for terminating the training process can be dictate by:
  - Time: Risk degradation in model performance
  - Threshold of prediction error/minimum accuracy with training data: Risk overfitting
- Cross-validation procedure to determine when to stop training
  - Save a portion of the data not used for training and testing
  - For example: with k-fold cross-validation, the training data is divided into k subsets (or "folds")
  - The model is trained on k-1 folds and validated on the remaining fold. This process is repeated k times, each time with a different fold used for validation. The model's performance is then averaged over these k iterations.



medium.com

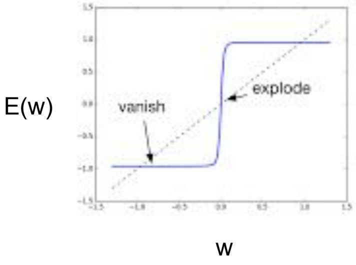
Regardless of the termination criteria used, the NN is not guaranteed to arrive at the global minimum for the SSE as it may become stuck in a local minimum which still represents a good , if not optimal solution

Wrapping up

- Once all the weights are updated, we complete one round of backpropagation
    - One forward pass followed by a backward pass are counted as one full pass
- With the updated weights, we then obtain the predicted output for the next data point, in another forward pass and compute the prediction error
- We repeat this until termination/stop criteria is reached which is termed that the model has converged

Vanishing Gradient Problem

- Vanishing Gradient Problem can occur during the training of NN when the gradient of the loss function wrt the weights in the lower layers of the network become very small
- Small gradient do not contribute much to the weight updates during training. As a result, it can slow or even halt learning those layers as the weights are not being updated effectively
- The problem is a result of the multiplicative effect of small derivatives of activation functions in deep networks



Examples of Activation Functions and Their Impact

- **Sigmoid Function**: It squashes its input to a range between 0 and 1. Its derivative is maximal at 0.25 and decreases toward 0 as the input moves away from 0. In deep networks, multiplying these small values can quickly lead to vanishing gradients.
- **Tanh Function**: Similar to sigmoid, but squashes input to a range between -1 and 1. It suffers from the same problem as sigmoid for high absolute values of input.
- **ReLU (Rectified Linear Unit)**: Introduced as a solution to the vanishing gradient problem. It does not saturate in the positive input range and has a derivative of either 0 (for negative inputs) or 1 (for positive inputs). However, ReLU can lead to another issue called the "dying ReLU problem," where neurons only output negative values and thus have a derivative of zero, effectively "dying."