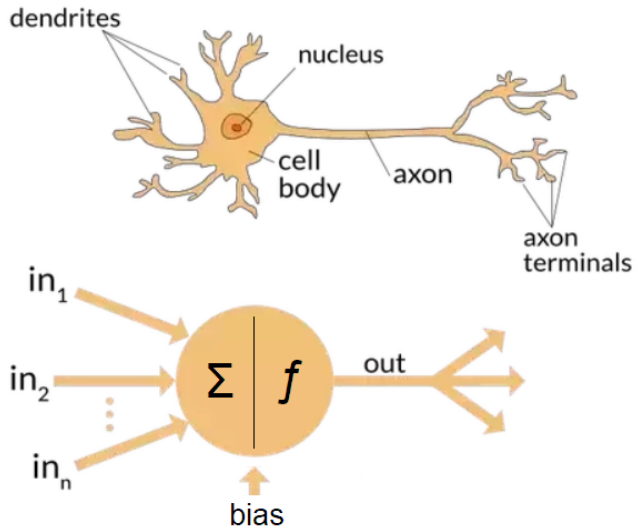
Back Propagation

Akhil Rana

December 19, 2018

Brief Introduction

- Artificial Neural Networks (ANN) concept has been inspired by biological neural network.
- First and foremost, neural network is a concept. It is not a machine or a physical box.
- In a biologic neural network, multiple neurons work together, receive input signals, process information and fire an output signal.

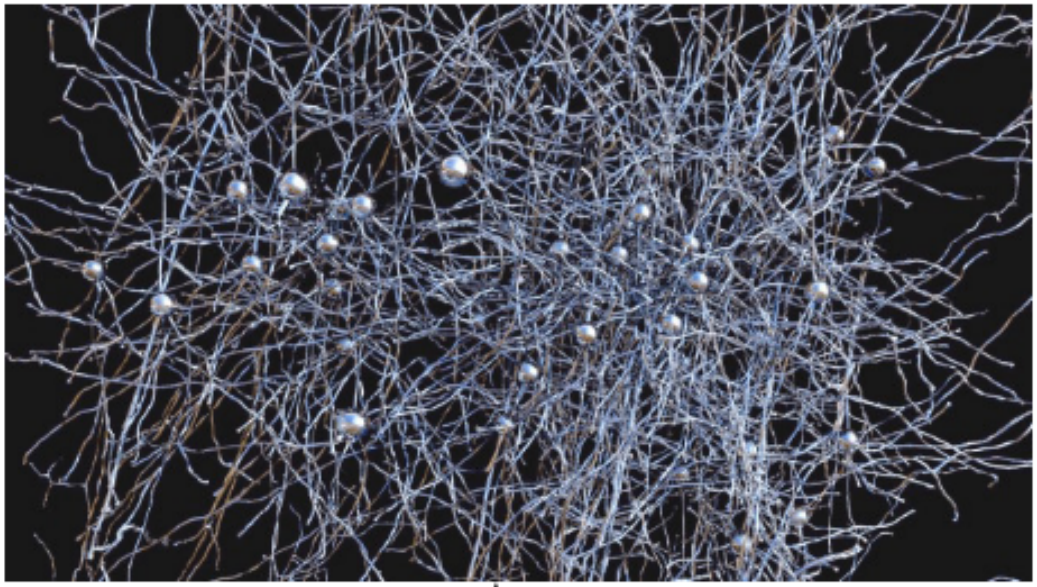


Biological Neuron Vs Artificial Neuron

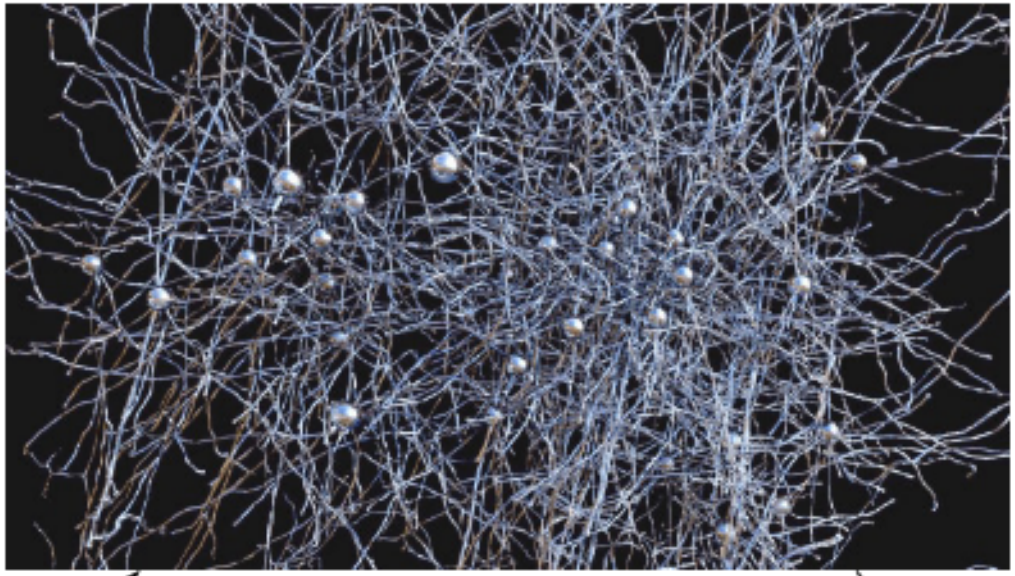
- Biological neurons are grouped in various layers and transmit updated signals. These signals contain information that can help us in determining patterns, identifying images, calculating numbers and making informed decisions throughout our life.
- AI neural network as a group of mathematical algorithms that produce expected output from the given input data.

Learning

Biological Forward Pass

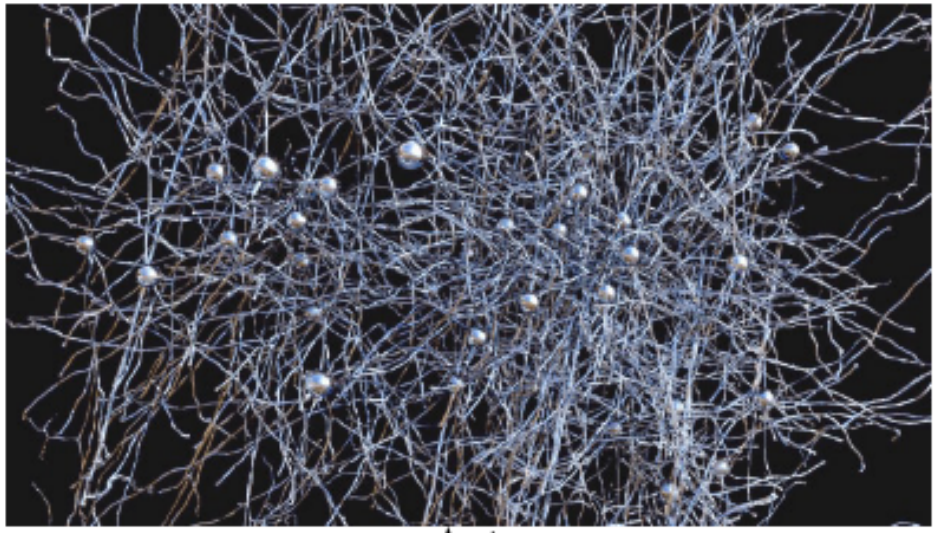


First Iteration

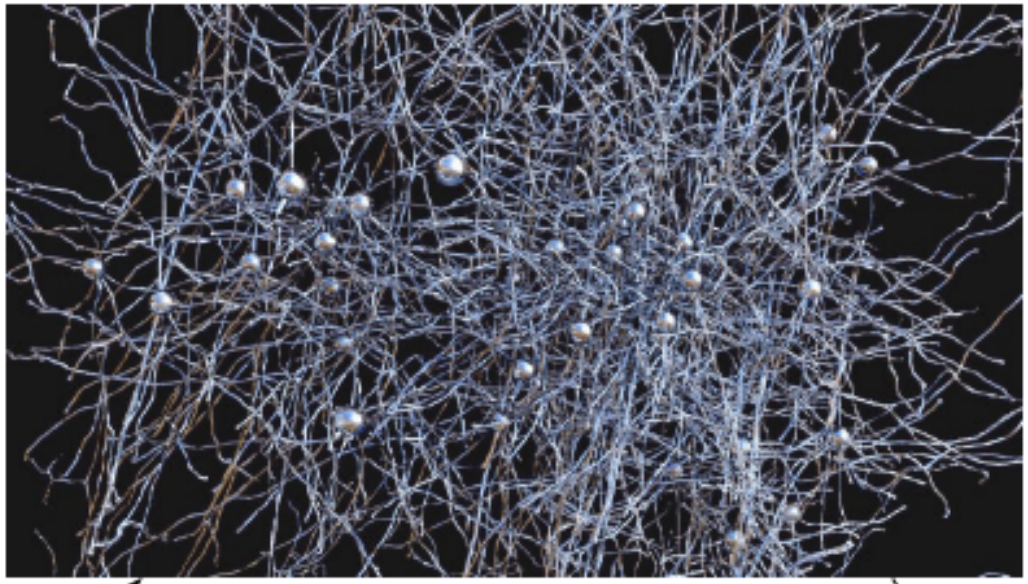


Lets Try Again

Biological Back Propagation



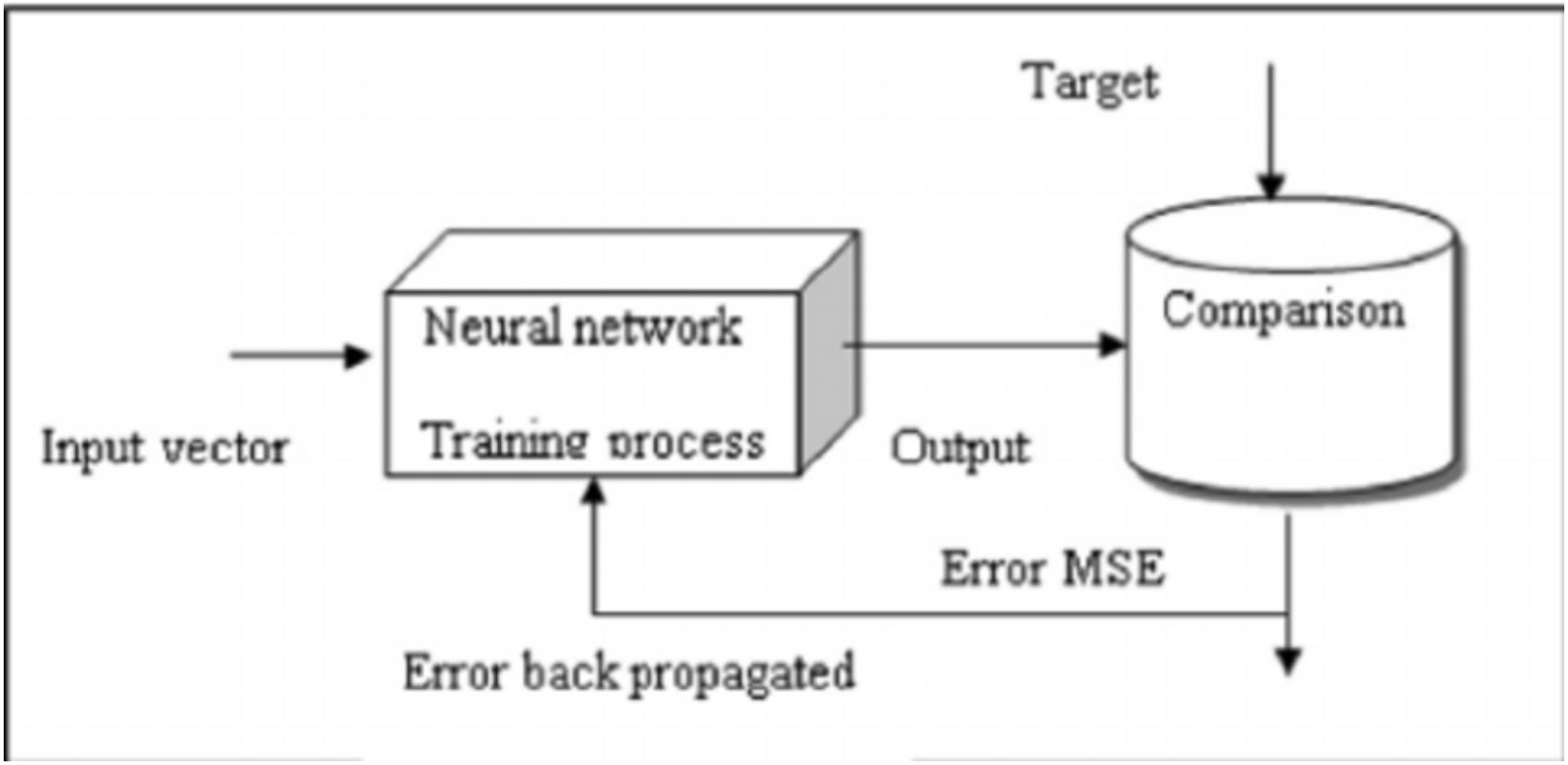
After few minutes



Back Propagation

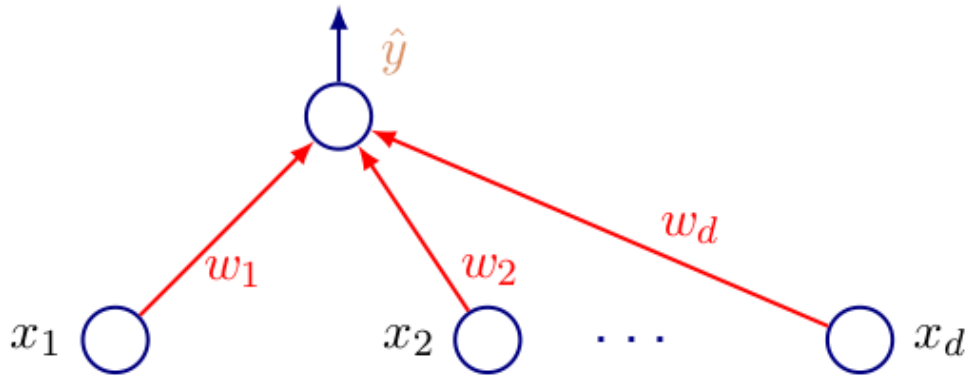
- In traditional software application (like conditional statements), a number of functions are coded. These functions take in inputs and produce an output. The inputs are not used to update the instructions.
- Neural networks are artificially intelligent. They can learn and improve themselves.
- Back propagation concept helps neural networks to improve their accuracy and makes Algorithms Self-Learning

Neural Network



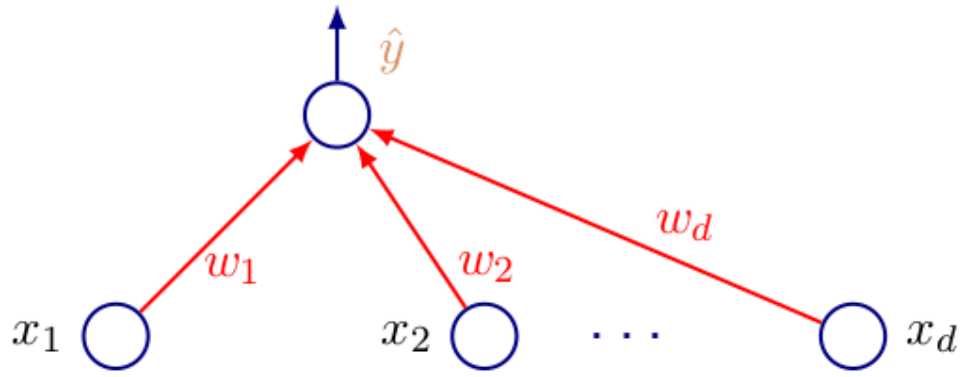
To understand, let us just calculate!

One Neuron Again



- Consider example $\mathbf{x}$; Output for $\mathbf{x}$ is $\hat{y}$; Correct Answer is y
- Loss $L = (y - \hat{y})^2$
- $\hat{y} = \mathbf{x}^T\mathbf{w} = x_1w_1 + x_2w_2 + \ldots x_dw_d$

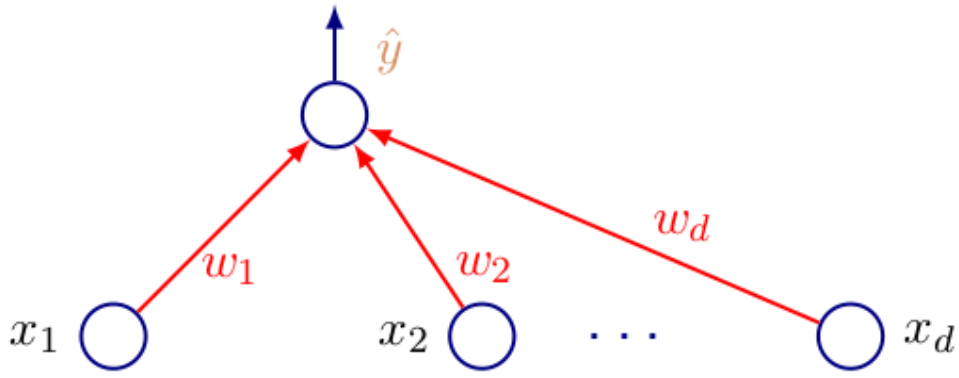
One Neuron Again



- Want to update w_i (forget closed form solution for a bit!)
- Update rule: $w_i := w_i - \eta \frac{\partial L}{\partial w_i}$
- Now

$$\frac{\partial L}{\partial w_i} = \frac{\partial (\hat{y} - y)^2}{\partial w_i} = 2(\hat{y} - y) \frac{\partial (x_1 w_1 + x_2 w_2 + \ldots x_d w_d)}{\partial w_i}$$

One Neuron Again

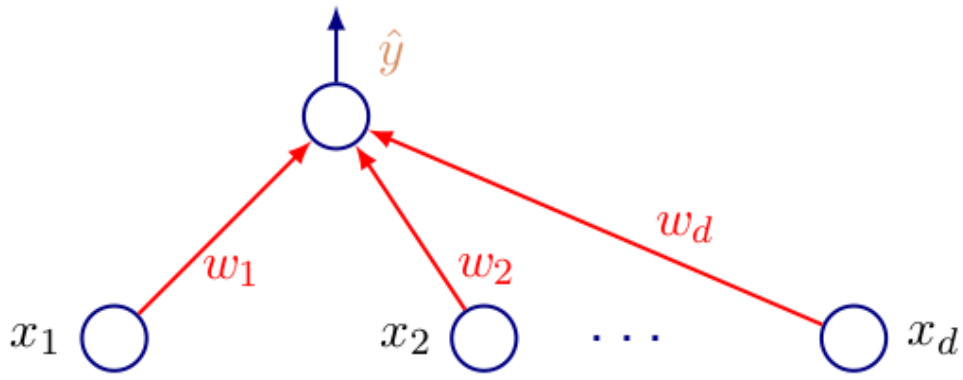


- We have: $\dfrac{\partial L}{\partial w_i} = 2(\hat{y} - y)x_i$
- Update Rule:

$$w_i := w_i - \eta(\hat{y} - y)x_i = w_i - \eta\delta x_i \text{ where } \delta = (\hat{y} - y)$$

- In vector form: $\mathbf{w} := \mathbf{w} - \eta\delta\mathbf{x}$
- Simple enough! Now let's graduate ...

One Neuron Again

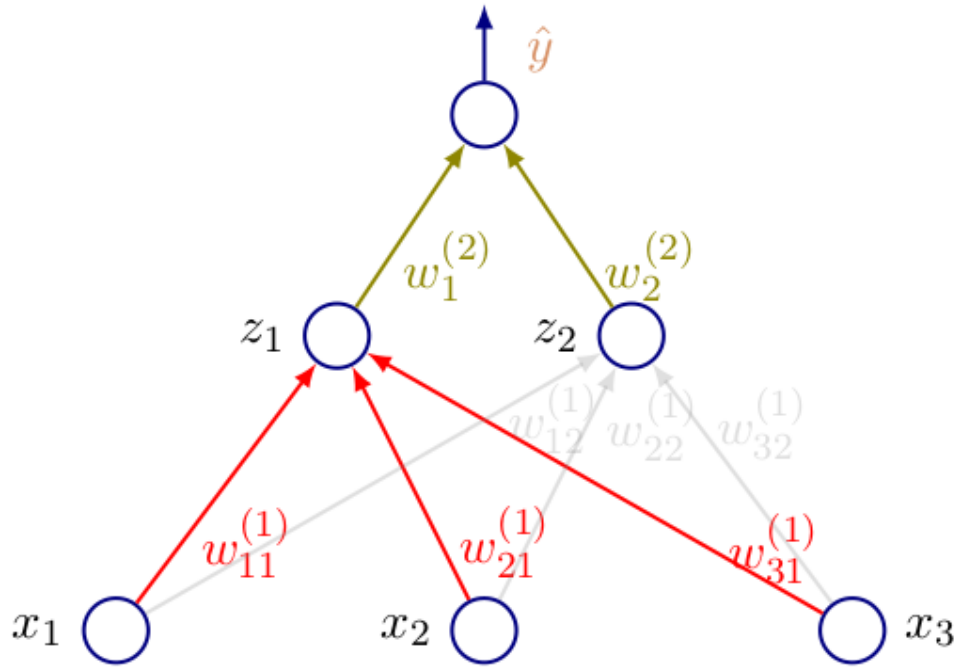


- We have: $\dfrac{\partial L}{\partial w_i} = 2(\hat{y} - y)x_i$
- Update Rule:

$$w_i := w_i - \eta(\hat{y} - y)x_i = w_i - \eta\delta x_i \text{ where } \delta = (\hat{y} - y)$$

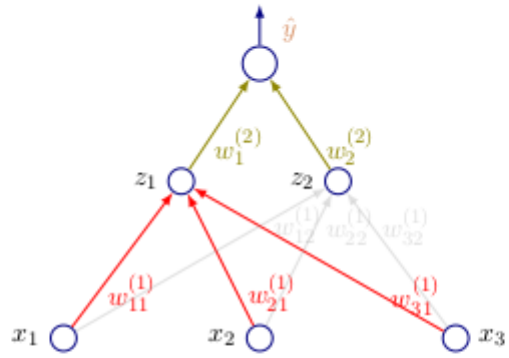
- In vector form: $\mathbf{w} := \mathbf{w} - \eta\delta\mathbf{x}$
- Simple enough! Now let's graduate ...

Simple Feedforward Network



- $\hat{y} = w_1^{(2)} z_1 + w_2^{(2)} z_2$
- $z_1 = \tanh(a_1)$ where $a_1 = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)} x_3$ likewise for z_2

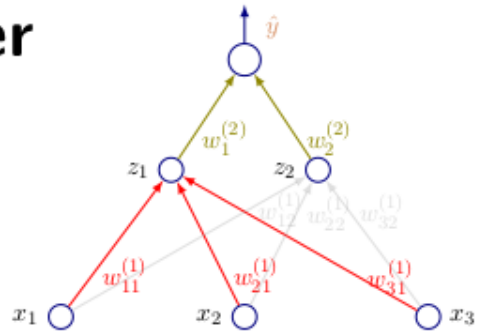
Simple Feedforward Network



- $z_1 = \tanh(a_1)$ where $a_1 = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)} x_3$
- $z_2 = \tanh(a_2)$ where $a_2 = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{32}^{(1)} x_3$

- Output $\hat{y} = w_1^{(2)} z_1 + w_2^{(2)} z_2$; Loss $L = (\hat{y} - y)^2$
- Want to assign credit for the loss L to each weight

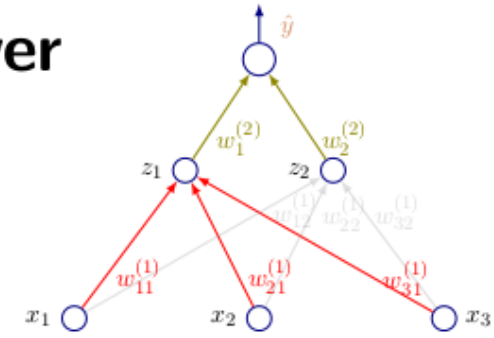
Top Layer



- Want to find: $\frac{\partial L}{\partial w_1^{(2)}}$ and $\frac{\partial L}{\partial w_2^{(2)}}$
- Consider $w_1^{(2)}$ first

- $\frac{\partial L}{\partial w_1^{(2)}} = \frac{\partial (\hat{y}-y)^2}{\partial w_1^{(2)}} = 2(\hat{y}-y)\frac{\partial (w_1^{(2)} z_1 + w_2^{(2)} z_2)}{\partial w_1^{(2)}} = 2(\hat{y}-y)z_1$
- Familiar from earlier! Update for $w_1^{(2)}$ would be $w_1^{(2)} := w_1^{(2)} - \eta \frac{\partial L}{\partial w_1^{(2)}} = w_1^{(2)} - \eta \delta z_1$ with $\delta = (\hat{y}-y)$
- Likewise, for $w_2^{(2)}$ update would be $w_2^{(2)} := w_2^{(2)} - \eta \delta z_2$

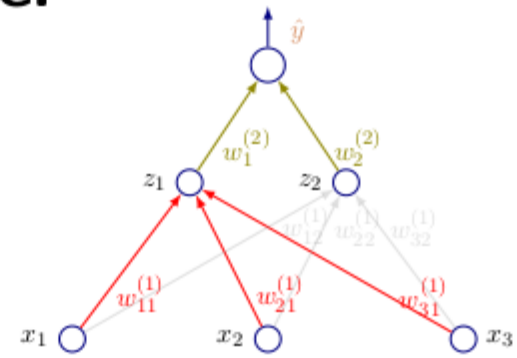
Next Layer



- There are six weights to assign credit for the loss incurred
- Consider $w_{11}^{(1)}$ for an illustration
- Rest are similar

- $\frac{\partial L}{\partial w_{11}^{(1)}} = \frac{\partial(\hat{y}-y)^2}{\partial w_{11}^{(1)}} = 2(\hat{y}-y)\frac{\partial(w_1^{(2)}z_1+w_2^{(2)}z_2)}{\partial w_{11}^{(21)}}$
- Now: $\frac{\partial(w_1^{(2)}z_1+w_2^{(2)}z_2)}{\partial w_{11}^{(1)}} = w_1^{(2)}\frac{\partial(\tanh(w_{11}^{(1)}x_1+w_{21}^{(1)}x_2+w_{31}^{(1)}x_3))}{\partial w_{11}^{(1)}} + 0$
- Which is: $w_1^{(2)}(1-\tanh^2(a_1))x_1$ recall $a_1 = ?$
- So we have: $\frac{\partial L}{\partial w_{11}^{(1)}} = 2(\hat{y}-y)w_1^{(2)}(1-\tanh^2(a_1))x_1$

Next Layer



- $\frac{\partial L}{\partial w_{11}^{(1)}} =$
  $2(\hat{y} - y)w_1^{(2)}(1 - \tanh^2(a_1))x_1$
- Weight update:
  $w_{11}^{(1)} := w_{11}^{(1)} - \eta \frac{\partial L}{\partial w_{11}^{(1)}}$

- Likewise, if we were considering $w_{22}^{(1)}$, we'd have:
- $\frac{\partial L}{\partial w_{22}^{(1)}} = 2(\hat{y} - y)w_2^{(2)}(1 - \tanh^2(a_2))x_2$
- Weight update: $w_{22}^{(1)} := w_{22}^{(1)} - \eta \frac{\partial L}{\partial w_{22}^{(1)}}$

Let's clean this up...

- Recall, for top layer: $\frac{\partial L}{\partial w_i^{(2)}} = (\hat{y} - y) z_i = \delta z_i$ (ignoring 2)
- One can think of this as: $\frac{\partial L}{\partial w_i^{(2)}} = \underbrace{\delta}_{\text{local error}} \underbrace{z_i}_{\text{local input}}$
- For next layer we had: $\frac{\partial L}{\partial w_{ij}^{(1)}} = (\hat{y} - y) w_j^{(2)} (1 - \tanh^2(a_j)) x_i$
- Let $\delta_j = (\hat{y} - y) w_j^{(2)} (1 - \tanh^2(a_j)) = \delta w_j^{(2)} (1 - \tanh^2(a_j))$ (Notice that δ_j contains the δ term (which is the error!))
- Then: $\frac{\partial L}{\partial w_{ij}^{(1)}} = \underbrace{\delta_j}_{\text{local error}} \underbrace{x_i}_{\text{local input}}$
- Neat!

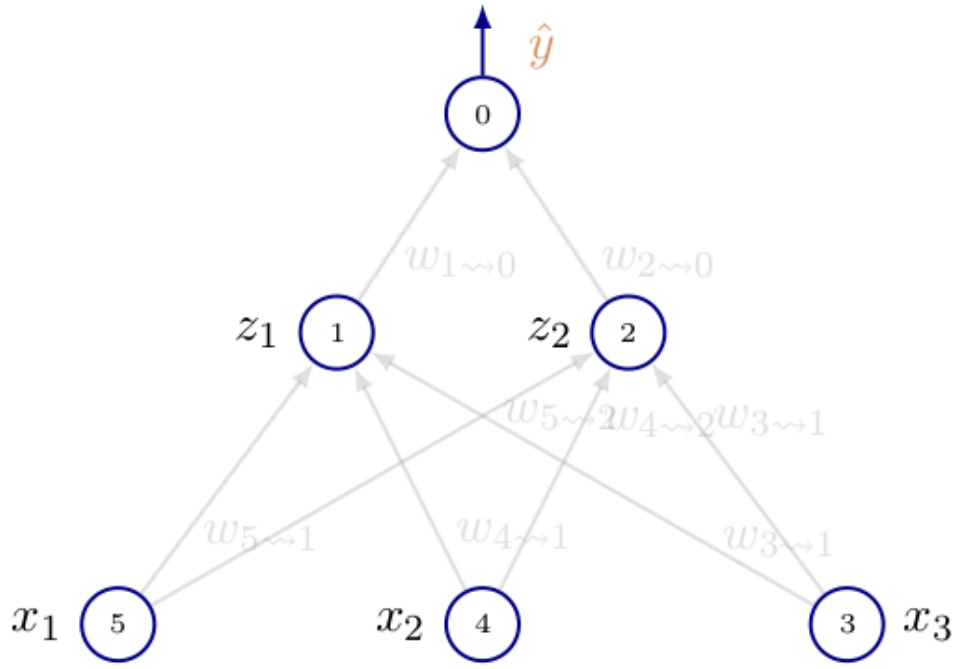
Let's clean this up...

- Let's get a cleaner notation to summarize this
- Let $w_{i \rightsquigarrow j}$ be the weight for the connection FROM node i to node j
- Then

$$\frac{\partial L}{\partial w_{i \rightsquigarrow j}} = \delta_j z_i$$

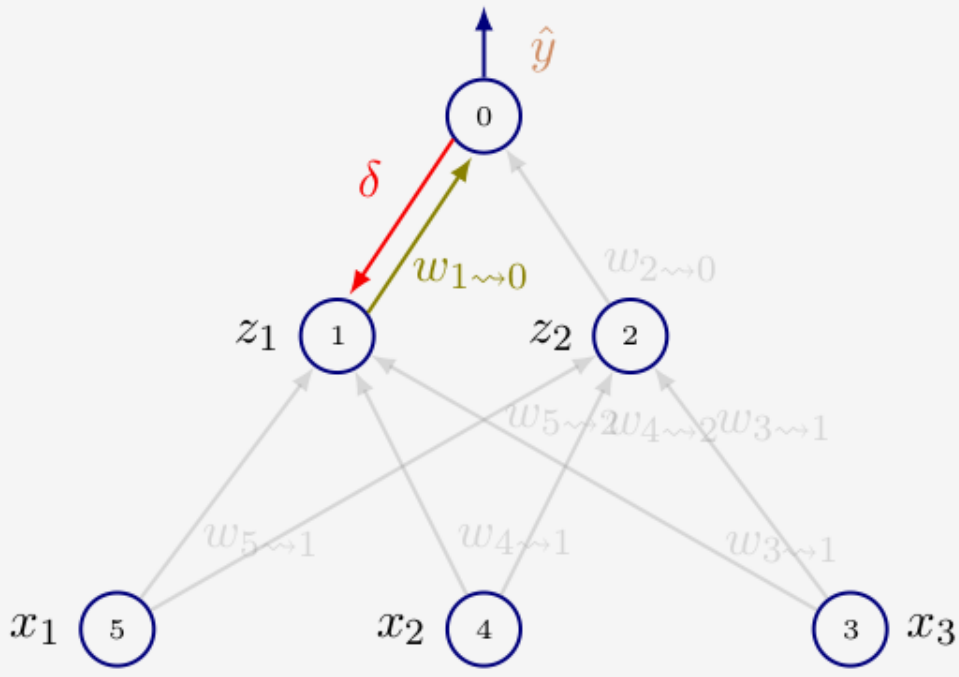
- δ_j is the local error (going from j backwards) and z_i is the local input coming from i

Credit Assignment: A Graphical Revision



- Let's redraw our toy network with new notation and label nodes

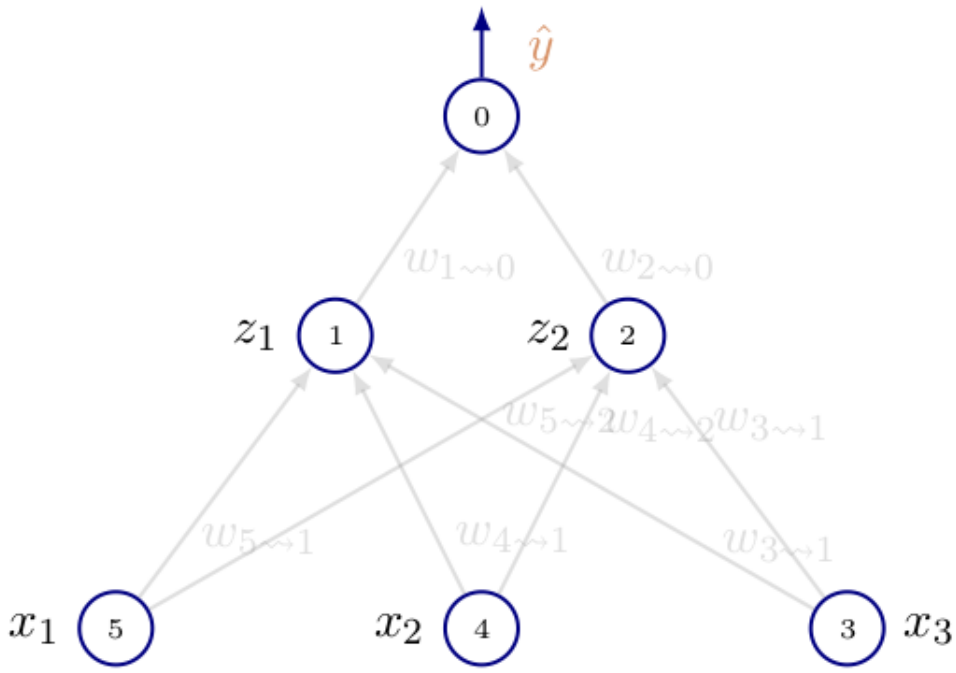
Credit Assignment: Top Layer



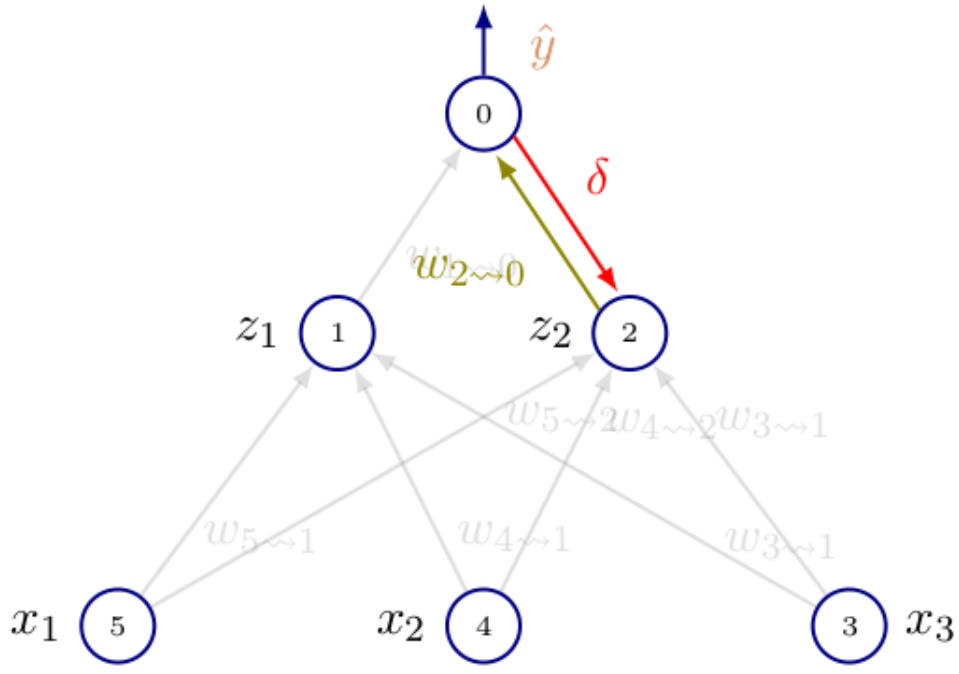
- Local error from 0: $\delta = (\hat{y} - y)$, local input from 1: z_1

$$\therefore \frac{\partial L}{\partial w_{1\rightsquigarrow 0}} = \delta z_1; \text{ and update } w_{1\rightsquigarrow 0} := w_{1\rightsquigarrow 0} - \eta \delta z_1$$

Credit Assignment: Top Layer



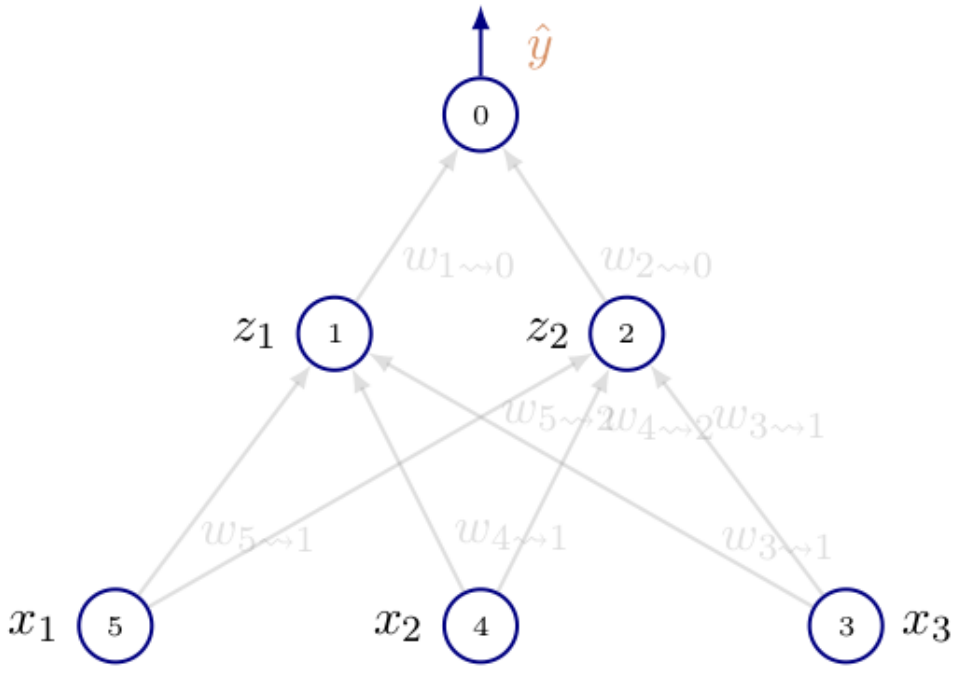
Credit Assignment: Top Layer



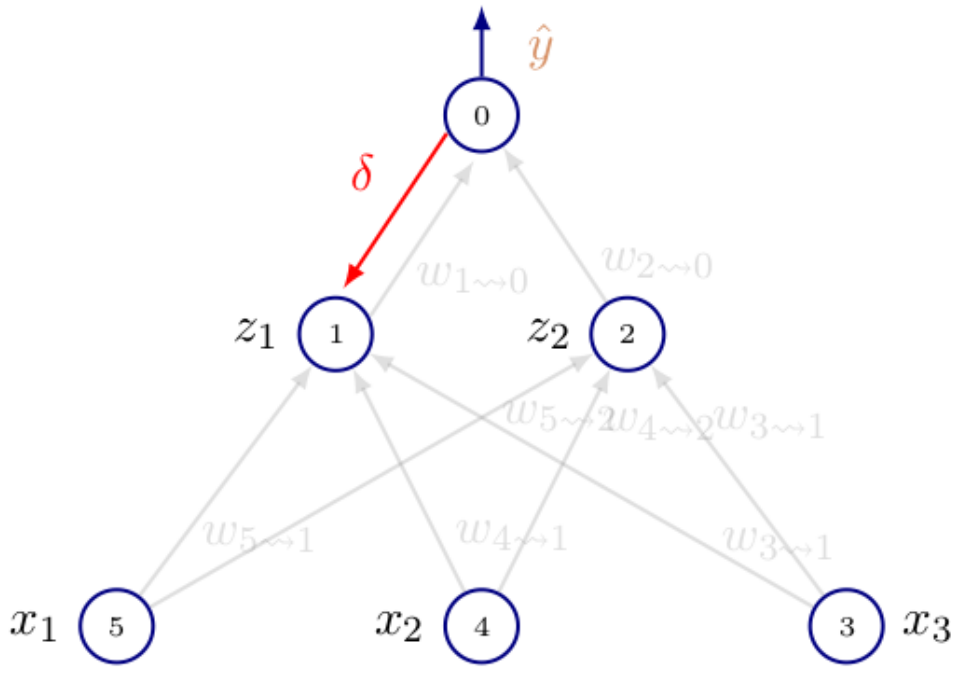
- Local error from 0: $\delta = (\hat{y} - y)$, local input from 2: z_2

$$\therefore \frac{\partial L}{\partial w_{2\rightsquigarrow 0}} = \delta z_2 \text{ and update } w_{2\rightsquigarrow 0} := w_{2\rightsquigarrow 0} - \eta \delta z_2$$

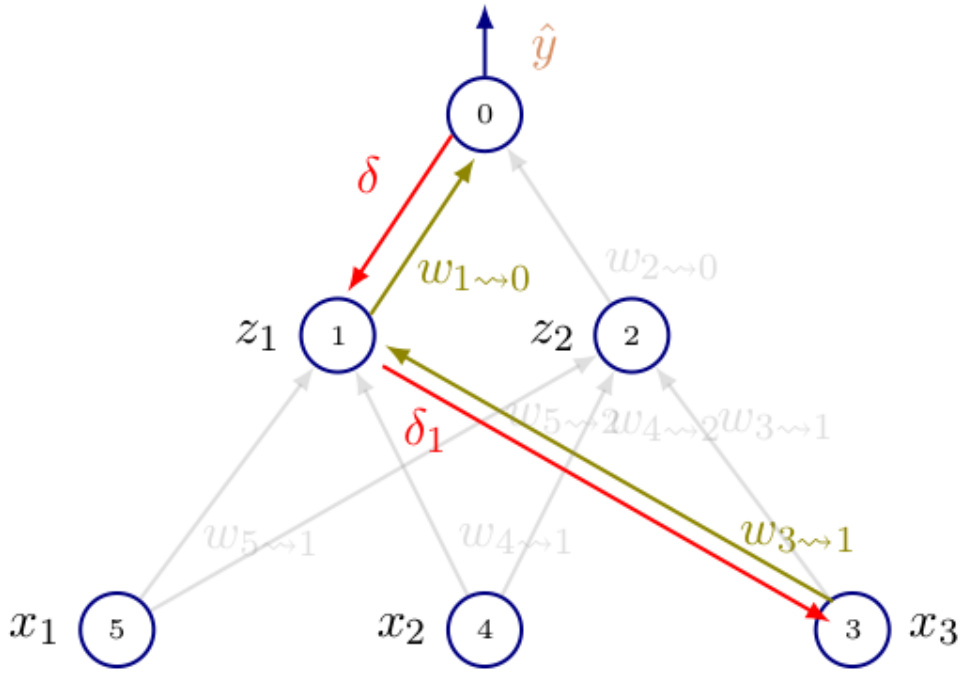
Credit Assignment: Next Layer



Credit Assignment: Next Layer



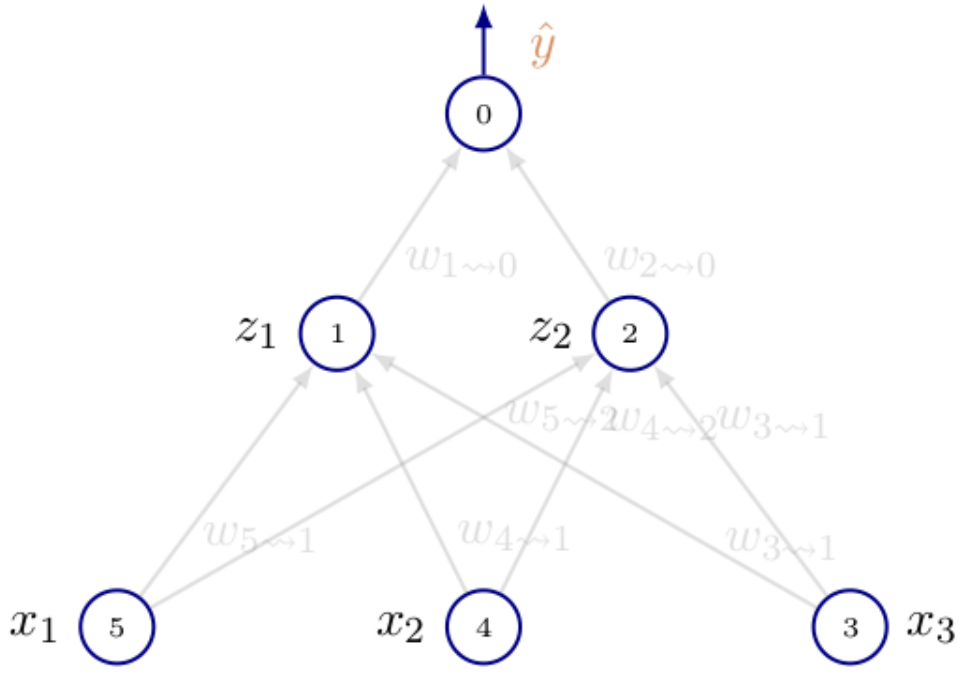
Credit Assignment: Next Layer



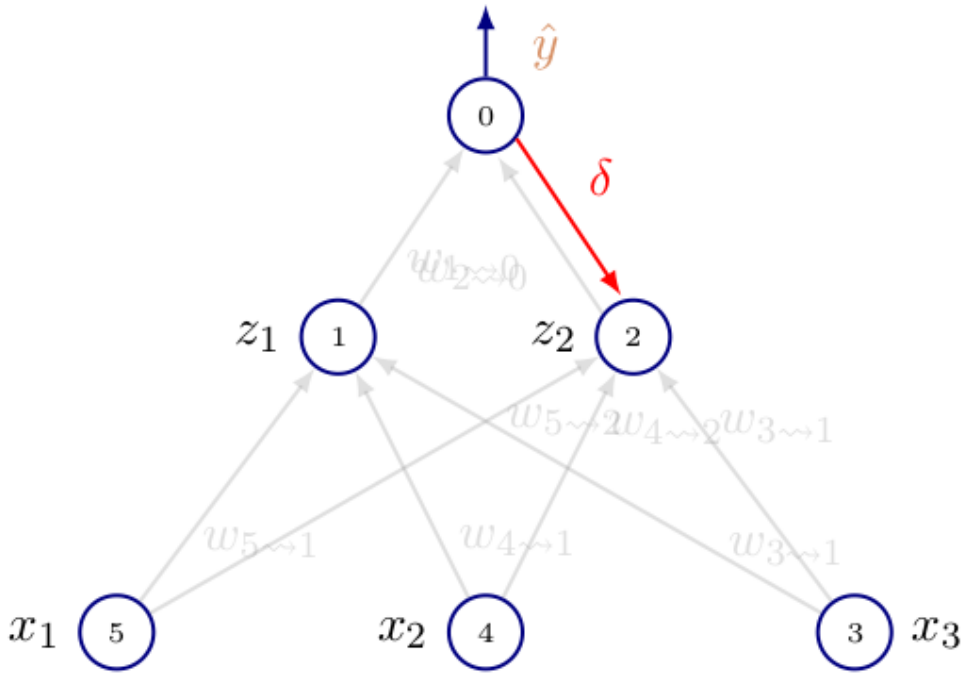
- Local error from 1: $\delta_1 = (\delta)(w_{1\rightsquigarrow 0})(1 - \tanh^2(a_1))$, local input from 3: x_3

$$\therefore \frac{\partial L}{\partial w_{3\rightsquigarrow 1}} = \delta_1 x_3 \text{ and update } w_{3\rightsquigarrow 1} := w_{3\rightsquigarrow 1} - \eta \delta_1 x_3$$

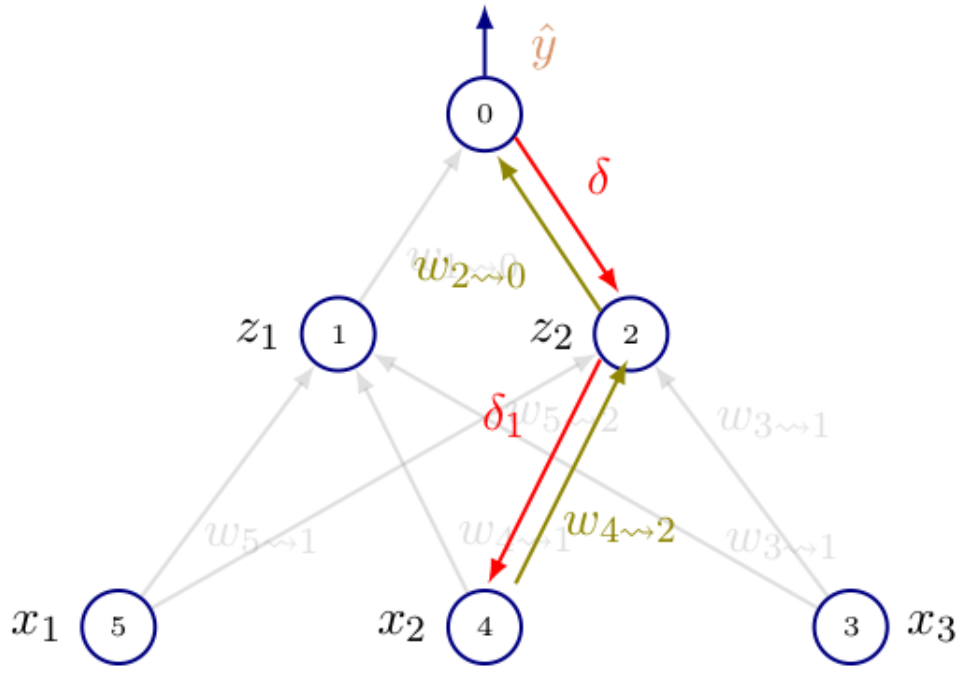
Credit Assignment: Next Layer



Credit Assignment: Next Layer



Credit Assignment: Next Layer



- Local error from 2: $\delta_2 = (\delta)(w_{2\rightsquigarrow 0})(1 - \tanh^2(a_2))$, local input from 4: x_2

$$\therefore \frac{\partial L}{\partial w_{4\rightsquigarrow 2}} = \delta_2 x_2 \text{ and update } w_{4\rightsquigarrow 2} := w_{4\rightsquigarrow 2} - \eta \delta_2 x_2$$

Let's Vectorize

- Let $W^{(2)} = \begin{bmatrix} w_{1 \rightsquigarrow 0} \\ w_{2 \rightsquigarrow 0} \end{bmatrix}$ (ignore that $W^{(2)}$ is a vector and hence more appropriate to use $\mathbf{w}^{(2)}$)
- Let

$$W^{(1)} = \begin{bmatrix} w_{5 \rightsquigarrow 1} & w_{5 \rightsquigarrow 2} \\ w_{4 \rightsquigarrow 1} & w_{4 \rightsquigarrow 2} \\ w_{3 \rightsquigarrow 1} & w_{3 \rightsquigarrow 2} \end{bmatrix}$$

- Let

$$Z^{(1)} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \text{ and } Z^{(2)} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

Feedforward Computation

1. Compute $A^{(1)} = {Z^{(1)}}^T W^{(1)}$
2. Applying element-wise non-linearity $Z^{(2)} = \tanh A^{(1)}$
3. Compute Output $\hat{y} = {Z^{(2)}}^T W^{(2)}$
4. Compute Loss on example $(\hat{y} - y)^2$

Flowing Backward

1. Top: Compute δ
2. Gradient w.r.t $W^{(2)} = \delta Z^{(2)}$
3. Compute $\delta_1 = ({W^{(2)}}^T \delta) \odot (1 - \tanh(A^{(1)})^2)$
   Notes: (a): $\odot$ is Hadamard product. (b) have written ${W^{(2)}}^T \delta$ as δ can be a vector when there are multiple outputs
4. Gradient w.r.t $W^{(1)} = \delta_1 Z^{(1)}$
5. Update $W^{(2)} := W^{(2)} - \eta \delta Z^{(2)}$
6. Update $W^{(1)} := W^{(1)} - \eta \delta_1 Z^{(1)}$
7. All the dimensionalities nicely check out!

So Far

- Backpropagation in the context of neural networks is all about assigning credit (or blame!) for error incurred to the weights
  - We follow the path from the output (where we have an error signal) to the edge we want to consider
  - We find the δs from the top to the edge concerned by using the chain rule
  - Once we have the partial derivative, we can write the update rule for that weight