The complete Mathematics for Machine Learning and Deep Learning

By: Adam Dhalla

1.3 Notation

m = Size of training set

n = Amount of input vars (ie. $x_1, x_2, \ldots, x_n$)

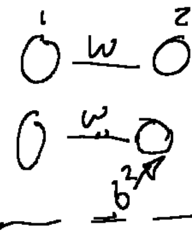
L = Amount of layers in the Neural Network

l = Specific layer (could be super/sub script)

W^l = Weights $\quad$ l = Layer going into

b^l = bias for layer l

x_i = Single input variable



Would be W^2

From 2.2 → Weights notation W^l_{jk} where j = node in layer l, k = node in layer $l-1$

See page 7 for example

1.5 Matrix Calculus Review

1.5.1 Gradients ($\mathbb{R}^n \to \mathbb{R}^1$)

For a function $F(x, y, \ldots)$ the gradient would be:

$$\begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \\ \vdots \end{bmatrix}$$

e.g. for a function $F(x, y) = x^2 + \cos(y)$, the gradient would be

$$\nabla F(x,y) = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix} = \begin{bmatrix} 2x \\ -\sin(y) \end{bmatrix}$$

☆ A nabla (∇) is the sign for gradient

1.5.2 Jacobians ($\mathbb{R}^n \to \mathbb{R}^m$)

Lets say we have a function that is $\mathbb{R}^2 \to \mathbb{R}^2$

$$F(x,y) = \begin{bmatrix} 2x + y^3 \\ e^y - 13x \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

where:

$$f_1 = 2x + y^3$$

$$f_2 = e^y - 13x$$

Then

$$\frac{\partial f_1}{\partial x} = 2 \qquad \frac{\partial f_1}{\partial y} = 3y^2$$

$$\frac{\partial f_2}{\partial x} = -13 \qquad \frac{\partial f_2}{\partial y} = e^y$$

$$J_F(x,y) = \begin{bmatrix} \nabla f_1^T \\ \nabla f_2^T \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} 2 & 3y^2 \\ -13 & e^y \end{bmatrix}$$

1.5.3 New way of seeing the scalar chain rule

Split a function into 2 functions

e.g. to find the derivative of $\sin(x^2)$ we can substitute g for x^2.

$$f = \sin(g) \qquad g = x^2$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} = \cos(g) \cdot 2x = 2x\cos(x^2)$$

1.5.4 Jacobian Chain rule

$$F(x,y) = \begin{bmatrix} \sin(x^2+y) \\ \ln(y^3) \end{bmatrix}$$

$$g = \begin{bmatrix} x^2+y \\ y^3 \end{bmatrix} \begin{array}{l} \leftarrow g_1 \\ \leftarrow g_2 \end{array}$$

$$F(x,y) = \begin{bmatrix} \sin(g_1) \\ \ln(g_2) \end{bmatrix}$$

$$\frac{\partial \vec{g}}{\partial \vec{x}} = \begin{bmatrix} 2x & 1 \\ 0 & 3y^2 \end{bmatrix}$$

$$\frac{\partial \vec{F}}{\partial \vec{g}} = \begin{bmatrix} \cos(g_1) & 0 \\ 0 & \frac{1}{g_2} \end{bmatrix}$$

$$\frac{\partial \vec{F}}{\partial \vec{x}} = \frac{\partial \vec{F}}{\partial \vec{g}}\frac{\partial \vec{g}}{\partial \vec{x}}$$

$$\frac{\partial \vec{F}}{\partial \vec{x}} = \begin{bmatrix} \cos(g_1) & 0 \\ 0 & \frac{1}{g_2} \end{bmatrix}\begin{bmatrix} 2x & 1 \\ 0 & 3y^2 \end{bmatrix}$$

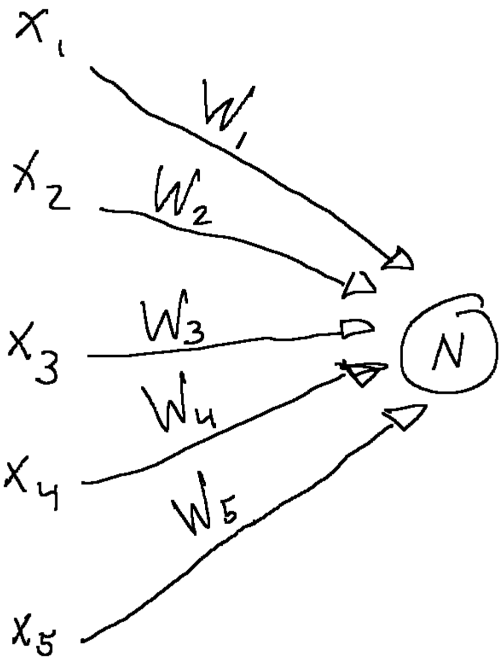
$$\frac{\partial \vec{F}}{\partial \vec{x}} = \begin{bmatrix} 2x\cos(g_1) & \cos(g_1) \\ 0 & \frac{3y^2}{g_2} \end{bmatrix}$$

$$\frac{\partial \vec{F}}{\partial \vec{x}} = \begin{bmatrix} 2x\cos(x^2+y) & \cos(x^2+y) \\ 0 & \frac{3\cancel{y^2}}{y^{\cancel{3}}} \end{bmatrix}$$

PART II:

Forward Propogation

2.1 The Neuron Function



$$\sigma\left(\overbrace{\overbrace{\sum_{i=1}^{n} x_i \, w_i}^{\text{Scalar}} + \overbrace{b}^{\text{scalar}}}^{\text{Scalar}}\right)$$

Activation Function inputs Weights bias

This can also be done by a dot product:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

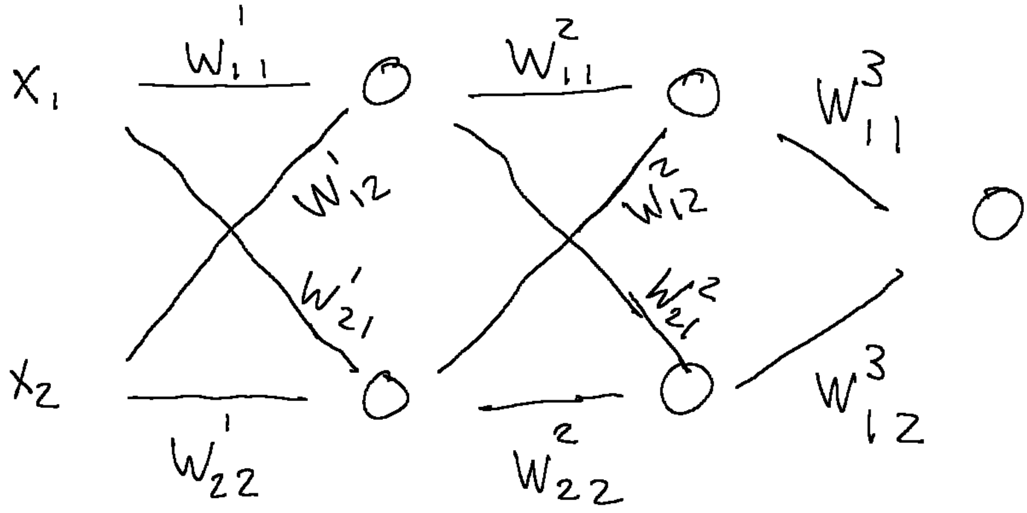
$$\sum_{i=1}^{n} x_i w_i = X^T W = x_1 w_1 + x_2 \; w_2 + \ldots + x_n w_n$$

$$\therefore$$

$$\sigma\big(\underbrace{X^T W + b}_{z}\big) \rightarrow \begin{aligned} z &= X^T W + b \\ a &= \sigma(z) \end{aligned}$$

A node is a Vector-to-Scalar function

2.2 Weight and Bias Indexing



Weights Notation : $\boxed{w^l_{jk}}$

where

l = Layer w is going into

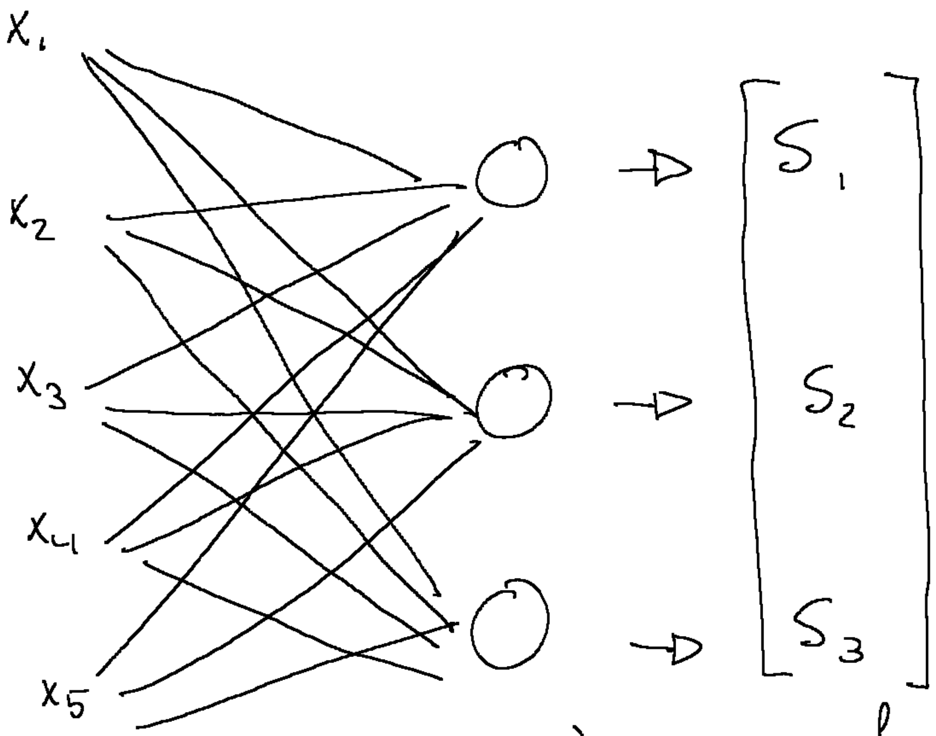
j = Node in layer l

K = Node in layer $l-1$

Bias Notation : b^l

note: Sometimes biases are attached to individual nodes (rather than an entire layer) where the notation would then be b^l_j

2.3 A layer of neurons



→ Another layer in the NN

$\sigma(x^T W + b)$ $\quad a^l$ → Activations of layer l

W^l → Matrix of weights (Remember w^l_{jk}, j^{th} row, k^{th} column)

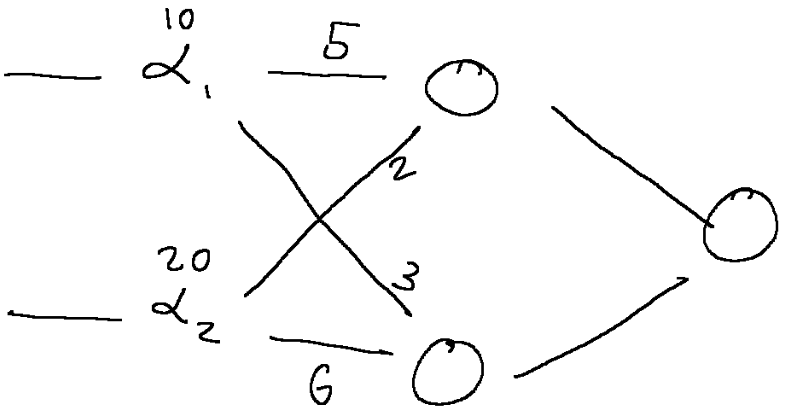
$$W = \left(\underbrace{\begin{bmatrix} W_{11} & W_{12} & \dots & W_{1k} \\ W_{21} & W_{22} & \dots & W_{2k} \\ \vdots & \vdots & & \vdots \\ W_{j1} & W_{j2} & \dots & W_{jk} \end{bmatrix}}_{k} \right) j$$

j ← Going into neuron position 'j' in layer l

k ↑ Coming from neuron position 'k' in layer $l-1$

2.3 cont...

Example of a weights matrix



$$W^1 = \begin{array}{cc} & k \\ \begin{bmatrix} \underset{(1,1)}{5} & \underset{(2,1)}{2} \\ \underset{(2,1)}{3} & \underset{(2,2)}{6} \end{bmatrix} & j \end{array}$$

Output of a layer:

$$\sigma(W^l a^{l-1} + b^l)$$

Activations of Prev layer

$$W^1 a^0 = \begin{bmatrix} 5 & 2 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \end{bmatrix} = \begin{bmatrix} 90 \\ 150 \end{bmatrix}$$

Then add bias. Lets say $b^1 = 6$

$$z^1 = \begin{bmatrix} 90 \\ 150 \end{bmatrix} + 6 = \begin{bmatrix} 96 \\ 156 \end{bmatrix}$$

Then we have the activation function

Lets say $\sigma = \text{ReLU}$

$$a^1 = \sigma(z^1) = \sigma\left(\begin{bmatrix} 96 \\ 156 \end{bmatrix}\right) = \begin{bmatrix} 96 \\ 156 \end{bmatrix}$$

In general, our weights matricies are (n, m) where n = row, m = col

n = nodes in layer l and m = nodes in layer $l-1$

Another way to represent a NN

$$a^0 \Rightarrow \sigma(w^1 a^0 + b^1) = a^1 \Rightarrow \sigma(w^2 a^1 + b^2) = a^2 \Rightarrow \ldots$$

PART III

Derivatives of Neural Networks and Gradient Descent

3.1 Motivation and Cost function

Cost function: A mathematical function that measures how well a neural network is Performing on the given data

Mean Square Error (MSE)

Training Examples

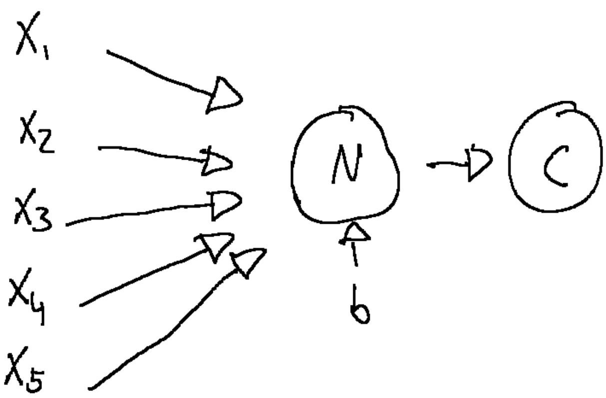
$$\frac{1}{2m}\sum_{i=1}^{m}\left(y-\hat{y}\right)^2$$

The 2 makes calculations easier w/ the derivative

Actual training example

Output of network

Simple Model



3.2 Differentiating a neuron's operations

3.2.1 Derivative of a binary element-wise operation

Binary element-wise operation: $F(\vec{v},\vec{w}) \rightarrow \vec{b}$

i.e. function that takes two vectors and returns a single vector where an operation is done 'element wise'.

e.g. $$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \underset{\text{Element-wise addition}}{\oplus} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} v_1 + w_1 \\ v_2 + w_2 \\ \vdots \\ v_n + w_n \end{bmatrix}$$

$$F(\vec{v},\vec{w}) = f(\vec{v}) \bigcirc g(\vec{w})$$

f: Does not need to be elementwise but can have a function to the vectors

$\bigcirc$: Element-wise

* An element wise operation can be a variety of things such as addition, multiplication and can even be a comparison (e.g '<') where a vector of binary elements is outputted

A lot of the time in element wise functions,

$$f(\vec{v}) = \vec{v}$$

$$g(\vec{w}) = \vec{w}$$

We can now take the jacobian of an element-wise operation

We are now passing through two vectors, and can get two jacobians out of this. One w.r.t. the elements of $\vec{v}$ and one w.r.t. the elements of $\vec{w}$

$\therefore$ we can find: $\dfrac{\partial F}{\partial \vec{w}}$ and $\dfrac{\partial F}{\partial \vec{v}}$

We will find the jacobian w.r.t. $\vec{v}$ $\left(\dfrac{\partial F}{\partial \vec{v}}\right)$

$$F(\vec{v}, \vec{w}) = \begin{array}{c} \begin{bmatrix} f_1(\vec{v}) \bigcirc g_1(\vec{w}) \\ f_2(\vec{v}) \bigcirc g_2(\vec{w}) \\ \vdots \\ f_n(\vec{v}) \bigcirc g_n(\vec{w}) \end{bmatrix} \end{array} \begin{array}{l} F_1 \\ F_2 \\ \\ F_n \end{array}$$

$$\frac{\partial F}{\partial \vec{v}} = \begin{bmatrix} \frac{\partial}{\partial v_1} f_1(\vec{v}) \bigcirc g_1(\vec{w}) & \frac{\partial}{\partial v_2} f_1(\vec{v}) \bigcirc g_1(\vec{w}) & \dots & \frac{\partial}{\partial v_n} f_1(\vec{v}) \bigcirc g_1(\vec{w}) \\ \frac{\partial}{\partial v_1} f_2(\vec{v}) \bigcirc g_2(\vec{w}) & \frac{\partial}{\partial v_2} f_2(\vec{v}) \bigcirc g_2(\vec{w}) & \dots & \frac{\partial}{\partial v_n} f_2(\vec{v}) \bigcirc g_2(\vec{w}) \\ & \vdots & & \\ \frac{\partial}{\partial v_1} f_n(\vec{v}) \bigcirc g_n(\vec{w}) & \frac{\partial}{\partial v_2} f_n(\vec{v}) \bigcirc g_n(\vec{w}) & \dots & \frac{\partial}{\partial v_n} f_n(\vec{v}) \bigcirc g_n(\vec{w}) \end{bmatrix}$$

if $f(\vec{v}) = \vec{v}$ and $g(\vec{w}) = \vec{w}$ then:

$$\frac{\partial F}{\partial \vec{v}} = \begin{bmatrix} \frac{\partial}{\partial v_1} v_1 \bigcirc w_1 & \frac{\partial}{\partial v_2} v_1 \bigcirc w_1 & \dots & \frac{\partial}{\partial v_n} v_1 \bigcirc w_1 \\ \frac{\partial}{\partial v_1} v_2 \bigcirc w_2 & \frac{\partial}{\partial v_2} v_2 \bigcirc w_2 & \dots & \frac{\partial}{\partial v_n} v_2 \bigcirc w_2 \\ & \vdots & & \\ \frac{\partial}{\partial v_1} v_n \bigcirc w_n & \frac{\partial}{\partial v_2} v_n \bigcirc w_n & \dots & \frac{\partial}{\partial v_n} v_n \bigcirc w_n \end{bmatrix}$$

The jacobian is very often a diagonal as $\frac{\partial}{\partial v_j}\left(f_i(\vec{v}) \bigcirc g_i(\vec{w})\right) = 0$

where $j \neq i$

as f_i and g_i are <u>not</u> functions of v_j

$\therefore \ \frac{\partial F}{\partial \vec{v}}$ simplifies to

$$\frac{\partial F}{\partial \vec{v}} = \begin{bmatrix} \frac{\partial}{\partial v_1} v_1 \bigcirc w_1, & 0 & , \ldots, & 0 \\ 0 & , \frac{\partial}{v_2} v_2 \bigcirc w_1 & , \ldots, & 0 \\ 0 & , \quad 0 & , \ldots, & \frac{\partial}{v_n} v_n \bigcirc w_n \end{bmatrix}$$

So for all elementwise functions, the jacobian will be diagonal.

This can simplify to

$$\frac{\partial F}{\partial \vec{v}} = \operatorname{diag}\left(\frac{\partial}{\partial v_1} v_1 \bigcirc w_1, \frac{\partial}{\partial v_2} v_2 \bigcirc w_2, \ldots \frac{\partial}{\partial v_n} v_n \bigcirc w_n\right)$$

and

$$\frac{\partial F}{\partial \vec{w}} = \operatorname{diag}\left(\frac{\partial}{\partial w_1} v_1 \bigcirc w_1, \frac{\partial}{\partial w_2} v_2 \bigcirc w_2, \ldots \frac{\partial}{\partial w_n} v_n \bigcirc w_n\right)$$

3.2.2. Derivative of a Hadamard Product

Hadamard Product: Element wise multiplies two vectors

$$F(\vec{V},\vec{w}) = \begin{bmatrix} f_1(\vec{V}) \otimes g_1(\vec{w}) \\ f_2(\vec{V}) \otimes g_2(\vec{w}) \\ \vdots \\ f_n(\vec{V}) \otimes g_n(\vec{w}) \end{bmatrix} = \begin{bmatrix} V_1 \otimes w_1 \\ V_2 \otimes w_2 \\ \vdots \\ V_n \otimes w_n \end{bmatrix} = \vec{V} \otimes \vec{w}$$

Since we are not manipulating $\vec{V}$ and $\vec{w}$ before element wise multiplying them, we can remap $f_n()$ and $g_n()$

$\therefore$

$$\frac{\partial F}{\partial \vec{V}} = \begin{bmatrix} \frac{\partial F_1}{\partial v_1} & \frac{\partial F_1}{\partial v_2} & \cdots & \frac{\partial F_1}{\partial v_n} \\ \frac{\partial F_2}{\partial v_1} & \frac{\partial F_2}{\partial v_2} & \cdots & \frac{\partial F_2}{\partial v_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial F_n}{\partial v_1} & \frac{\partial F_n}{\partial v_2} & \cdots & \frac{\partial F_n}{\partial V_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial v_1} & 0 & \cdots & 0 \\ 0 & \frac{\partial f_2}{\partial F_2} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \frac{\partial f_n}{\partial V_n} \end{bmatrix}$$

The derivative, $\frac{\partial F_n}{\partial V_n}(V_n \otimes W_n) = W_n$, so:

$$\frac{\partial F}{\partial \vec{V}} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix}$$

Equivalently:

$$\frac{\partial F}{\partial \vec{w}} = \begin{bmatrix} V_1 & 0 & \cdots & 0 \\ 0 & V_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & V_n \end{bmatrix}$$

3.2.3 Derivative of a Scalar Expansion

Scalar Expansion: Multiplying a scalar w/ a Vector

$$2 \cdot \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ \vdots \\ 2 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} 2V_1 \\ 2V_2 \\ \vdots \\ 2V_n \end{bmatrix}$$

We broadcast/expand the scalar to be a vector of the same size

$$F(\vec{V}, \underbrace{x}_{\text{scalar}}) = f(\vec{V}) \circ g(x)$$

$$\text{where } g(x) = \vec{1} \cdot x$$

(The act of multiplying x by the ones Vector is an act of broadcasting itself)

Expands x to be a Vector by multiplying it by the ones vector

$$F(\vec{V}, x) = \begin{bmatrix} f_1(\vec{V}) \circ g_1(x) \\ f_2(\vec{V}) \circ g_2(x) \\ \vdots \\ f_n(\vec{V}) \circ g_n(x) \end{bmatrix}$$

$$\frac{\partial F}{\partial \vec{V}} = \begin{bmatrix} \frac{\partial f_1}{\partial V_1} & \frac{\partial f_1}{\partial V_2} & \cdots & \frac{\partial f_1}{\partial V_n} \\ \frac{\partial f_2}{\partial V_1} & \frac{\partial f_2}{\partial V_2} & \cdots & \frac{\partial f_2}{\partial V_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial V_1} & \frac{\partial f_n}{\partial V_2} & \cdots & \frac{\partial f_n}{\partial V_n} \end{bmatrix} = \begin{bmatrix} x & 0 & \cdots & 0 \\ 0 & x & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & x \end{bmatrix}$$

Since x is a scalar, taking the derivative w.r.t. x will give us a gradient not a jacobian.

If elementwise is multiplication then:

$$\nabla f_x = \begin{bmatrix} \frac{\partial f_1}{\partial x} \\ \frac{\partial f_2}{\partial x} \\ \vdots \\ \frac{\partial f_n}{\partial x} \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

If elementwise is addition then

$$\nabla f_x = \begin{bmatrix} \frac{\partial f_1}{\partial x} \\ \frac{\partial f_2}{\partial x} \\ \vdots \\ \frac{\partial f_n}{\partial x} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

If elementwise is subtraction then

$$\nabla f_x = \begin{bmatrix} \frac{\partial f_1}{\partial x} \\ \frac{\partial f_2}{\partial x} \\ \vdots \\ \frac{\partial f_n}{\partial x} \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$$

3.2.4. Derivative of a sum

$$\vec{V} = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} = \sum_{i=1}^{n} g_i(V)$$

A summation is a vector-to-scalar function

The derivative of the sum:

$$\frac{\partial s}{\partial \vec{V}} = \begin{bmatrix} \frac{\partial s}{V_1} & \frac{\partial s}{V_2} & \dots & \frac{\partial s}{V_n} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial}{\partial V_1} \sum_{i=1}^{n} g_i(V) & \frac{\partial}{\partial V_2} \sum_{i=1}^{n} g_i(V) & \dots & \frac{\partial}{\partial V_n} \sum_{i=1}^{n} g_i(V) \end{bmatrix}$$

The derivative of a sum is equivalent to the sum of a derivative

:

$$= \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial}{\partial V_1} g_i(V) & \sum_{i=1}^{n} \frac{\partial}{\partial V_2} g_i(V) & \dots & \sum_{i=1}^{n} \frac{\partial}{\partial V_n} g_i(V) \end{bmatrix}$$

If $g(v) = v$

$$= \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial}{\partial V_1} V_i & \sum_{i=1}^{n} \frac{\partial}{\partial V_2} V_i & \dots & \sum_{i=1}^{n} \frac{\partial}{\partial V_n} V_i \end{bmatrix}$$

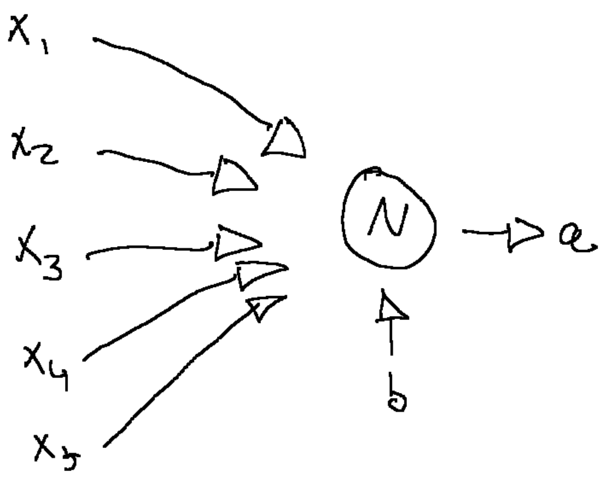
The derivative of v, for every Summation, will be 0 everywhere except on the element, v_i, being derived where it would be 1. So

$$= \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$$

If instead $g(v) = z \cdot v$, then

$$= \begin{bmatrix} z & z & \dots & z \end{bmatrix}$$

3.3 Derivative of a neuron's activation



$$a = \sigma(W^T x + b) = \sigma(z)$$

where $z = W^T x + b$

We can further break this down

$$a = \sigma(W^T x + b) = \sigma\Big(\underbrace{\text{sum}\big(\underbrace{W \otimes X}_{H}\big)}_{S(H)} + b\Big)$$

We will now investigate the derivative of the weights and bias using the chain rule

$$\text{weights: } \frac{\partial a}{\partial w} = \frac{\partial a}{\partial z} \frac{\partial z}{\partial w} = \frac{\partial a}{\partial z} \frac{\partial z}{\partial S} \frac{\partial S}{\partial H} \frac{\partial H}{\partial w}$$

$$\text{Bias: } \frac{\partial a}{\partial b} = \frac{\partial a}{\partial z} \frac{\partial z}{\partial b}$$

We will solve for this on the next 2 pages

We can now find $\frac{\partial a}{\partial w}$

$$\frac{\partial a}{\partial w} = \frac{\partial a}{\partial z}\frac{\partial z}{\partial w} = \frac{\partial a}{\partial z}\frac{\partial z}{\partial S}\frac{\partial S}{\partial H}\frac{\partial H}{\partial w}$$

$$\frac{\partial H}{\partial w} = \frac{\partial}{\partial w}(W \otimes X) = \begin{bmatrix} x_1 & 0 & \dots & 0 \\ 0 & x_2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & x_n \end{bmatrix} = \mathrm{diag}(x_1 \dots x_n)$$

So:

$$\frac{\partial a}{\partial w} = \frac{\partial a}{\partial z}\frac{\partial z}{\partial S}\frac{\partial S}{\partial H}\frac{\partial H}{\partial w} = \frac{\partial a}{\partial z}\frac{\partial z}{\partial S}\frac{\partial S}{\partial H}\mathrm{diag}(x_1 \dots x_n)$$

$$\frac{\partial S}{\partial H} = \frac{\partial}{\partial H}\mathrm{Sum}(X \otimes W) = \vec{1}^T$$

note: $\vec{1}^T \cdot \mathrm{diag}(x_1 \dots x_n) = X^T$

So:

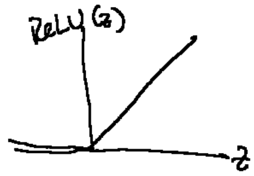
$$\frac{\partial a}{\partial w} = \frac{\partial a}{\partial z}\frac{\partial z}{\partial S}\frac{\partial S}{\partial H}\mathrm{diag}(x_1 \dots x_n) = \frac{\partial a}{\partial z}\frac{\partial z}{\partial S}\vec{1}^T\mathrm{diag}(x_1 \dots x_n) = \frac{\partial a}{\partial z}\frac{\partial z}{\partial S}X^T$$

$$\frac{\partial z}{\partial S} = \frac{\partial}{\partial S}\left(S(H) + b\right) = 1$$

So

$$\frac{\partial a}{\partial w} = \frac{\partial a}{\partial z}\frac{\partial z}{\partial S}X^T = \frac{\partial a}{\partial z} \cdot 1 \cdot X^T = \frac{\partial a}{\partial z}X^T = \frac{\partial a}{\partial z}[x_1, x_2, \dots, x_n]$$

Using $\mathrm{ReLU} = \max(0, z)$



The derivative of ReLU is:

$$\frac{\partial a}{\partial z} = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$

$$\frac{\partial a}{\partial w} = \frac{\partial a}{\partial z}X^T = \begin{cases} \vec{0}^T & \text{if } W^T x + b \leq 0 \\ \vec{X}^T & \text{if } W^T x + b > 0 \end{cases}$$

We can now find $\frac{\partial a}{\partial b}$

$$\frac{\partial a}{\partial b} = \frac{\partial a}{\partial z} \frac{\partial z}{\partial b}$$

$$\frac{\partial z}{\partial b} = 1$$

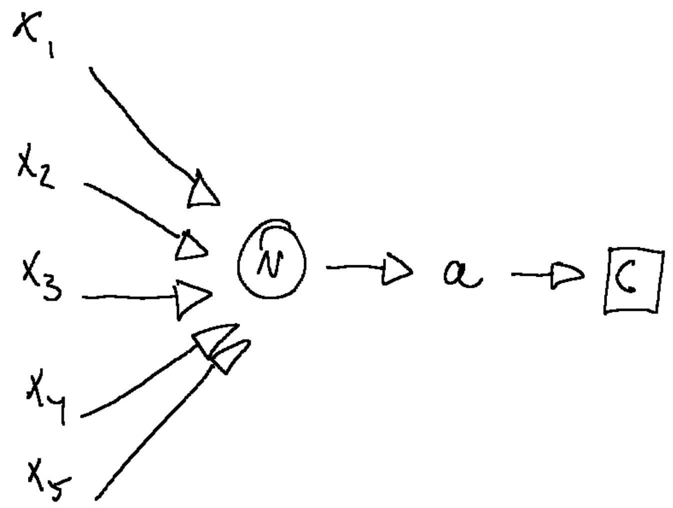
so

$$\frac{\partial a}{\partial b} = \frac{\partial a}{\partial z} \frac{\partial z}{\partial b} = \frac{\partial a}{\partial z} \cdot 1 = \frac{\partial a}{\partial z}$$

Thus

$$\frac{\partial a}{\partial b} = \frac{\partial a}{\partial z} = \begin{cases} 0 & \text{if } W^T x + b \leq 0 \\ 1 & \text{if } W^T x + b > 0 \end{cases}$$

3.4 Derivative of the cost for a simple neural network



Cost (c) is Mean Square Error

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} (y - \hat{y})^2$$

where

$\hat{y}$ = activations = last layer of activations

We want to find $\frac{\partial c}{\partial w}$ and $\frac{\partial c}{\partial b}$

$$\frac{\partial c}{\partial w} = \frac{\partial c}{\partial a} \frac{\partial a}{\partial w} \qquad \text{where } \frac{\partial a}{\partial w} = \begin{cases} \vec{0}^T & \text{if } W^T X + b \leq 0 \\ \frac{\partial z}{\partial w} & \text{if } W^T X + b > 0 \end{cases}$$

$$\frac{\partial c}{\partial b} = \frac{\partial c}{\partial a} \frac{\partial a}{\partial b} \qquad \text{where } \frac{\partial a}{\partial b} = \begin{cases} 0 & \text{if } w^T x + b \leq 0 \\ 1 & \text{if } W^T X + b > 0 \end{cases}$$

Training example is a matrix where each training example is a column

$$X = \begin{bmatrix} X_1^1 & X_1^2 & \dots & X_1^m \\ X_2^1 & X_2^2 & \dots & X_2^m \\ \vdots & \vdots & & \vdots \\ X_n^1 & X_n^2 & \dots & X_n^m \end{bmatrix}$$

$$\frac{1}{2m}\sum_{i=1}^{m}\left(y-a^L\right)=\frac{1}{2m}\sum_{i=1}^{m}V^2$$

where

$$\left(y-a^L\right)=V$$

The chain rule then becomes:

$$\frac{\partial c}{\partial w}=\frac{\partial c}{\partial V}\frac{\partial V}{\partial a}\frac{\partial a}{\partial w}=\frac{\partial c}{\partial V}\frac{\partial V}{\partial w}$$

$$\frac{\partial V}{\partial w}=\frac{\partial}{\partial w}\left(y-a^L\right)=\frac{\partial}{\partial w}\left(0-a^L\right)=-\frac{\partial a^L}{\partial w}=\frac{\partial a}{\partial w}$$

We now get

$$\frac{\partial c}{\partial w}=\frac{\partial c}{\partial V}\frac{\partial V}{\partial w}=\frac{\partial c}{\partial V}-\frac{\partial a}{\partial w}$$

Lets find $\frac{\partial c}{\partial V}$

$$\frac{\partial c}{\partial w}=\frac{\partial}{\partial w}\frac{1}{2m}\sum_{i=1}^{m}V^2=\frac{1}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial w}V^2$$

$$=\frac{1}{2m}\sum_{i=1}^{m}\frac{\partial V^2}{\partial V}\frac{\partial V}{\partial w}=\frac{1}{2m}\sum_{i=1}^{m}2V\cdot-\frac{\partial a}{\partial w}$$

$$=\frac{1}{m}\sum_{i=1}^{m}V\cdot-\frac{\partial a}{\partial w}=\frac{1}{m}\sum_{i=1}^{m}V\begin{cases}\vec{0}^T & \text{if } w^Tx+b\le 0\\ -\frac{\partial z}{\partial w} & \text{if } w^Tx+b>0\end{cases}$$

$$=\frac{1}{m}\begin{cases}\vec{0}^T & \text{if } w^Tx+b\le 0\\ -V\frac{\partial z}{\partial w} & \text{if } w^Tx+b>0\end{cases}=\frac{1}{m}\begin{cases}\vec{0}^T & \text{if } w^Tx+b\le 0\\ -V\cdot x^T & \text{if } w^Tx+b>0\end{cases}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \begin{cases} \vec{0}^T & \text{if } x^T w + b \leq 0 \\ -(y - a^L) x^T & \text{if } x^T w + b > 0 \end{cases}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \begin{cases} \vec{0}^T & \text{if } x^T w + b \leq 0 \\ -(y - \underbrace{\max(0, w^T x + b)}_{\text{This represents ReLU}}) x^T & \text{if } x^T w + b > 0 \end{cases}$$

The piece wise and the $\max(0, w^T x + b)$ do the same thing.

$$\therefore = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} \vec{0}^T & \text{if } x^T w + b \leq 0 \\ -(y - (w^T x + b)) \cdot x^T & \text{if } x^T w + b > 0 \end{cases}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \begin{cases} \vec{0}^T & \text{if } x^T w + b \leq 0 \\ (w^T x + b - y) \cdot x^T & \text{if } x^T w + b > 0 \end{cases}$$

$$\boxed{\frac{\partial L}{\partial w} = \frac{1}{m} \begin{cases} \vec{0}^T & \text{if } x^T w + b \leq 0 \\ \sum_{i=1}^{m} (w^T x + b - y) x^T & \text{if } x^T w + b > 0 \end{cases}}$$

3.5 Understanding the derivative of the Cost w.r.t. the weights

$$\frac{\partial c}{\partial w} = \frac{1}{m} \begin{cases} \vec{0}^T & \text{if } w^T x + b \leq 0 \\ \sum_{i=1}^{m} (w^T x + b - y) x^T & \text{if } w^T x + b > 0 \end{cases}$$

We want:

$$\begin{bmatrix} \frac{\partial c}{\partial w_1} \\ \frac{\partial c}{\partial w_2} \\ \vdots \\ \frac{\partial c}{\partial w_n} \end{bmatrix}$$

i.e. how does the cost change when we change one weight. We use this for gradient descent to know how much/what to change the specific weight by

---

$w^T x + b - y = a^L - y_i$ is the error term (e_i)

$\therefore$

$$\frac{\partial c}{\partial w} = \frac{1}{m} \begin{cases} \vec{0}^T & \text{if } w^T x + b \leq 0 \\ \sum_{i=1}^{m} e_i x^T & \text{if } w^T x + b > 0 \end{cases}$$

The error term, e_i, is how incorrect the answer is.

∴ For every training example, i, we get:

$$\frac{\partial c}{\partial \vec{w}} = e_i X^T = \begin{bmatrix} e_i X_1 \\ e_i X_2 \\ \vdots \\ e_i X_n \end{bmatrix} \begin{array}{l} \rightarrow \frac{\partial c}{\partial w_1} \\ \rightarrow \frac{\partial c}{\partial w_2} \\ \\ \rightarrow \frac{\partial c}{\partial w_n} \end{array}$$

The $\frac{\partial c}{\partial \vec{w}}$ vector represents the ratios of change in C when changing the weights by some amount

---

For multiple training examples:

For each training example, X^T is the same but there is a different error, e_i

$$\frac{\partial c}{\partial \vec{w}} = \frac{1}{m} \sum_{i=1}^{m} e_i X^T = \frac{1}{m} \cdot \begin{bmatrix} e_1 X_1 + e_2 X_1 + \ldots e_m X_1 \\ e_1 X_2 + e_2 X_2 + \ldots e_m X_2 \\ \vdots \qquad \vdots \qquad \vdots \\ e_1 X_n + e_2 X_n + \ldots e_m X_n \end{bmatrix}$$

Each row is an approximate derivative of the cost w.r.t. all weights averaged $\left(\frac{1}{m}\right)$ over all training example

3.6 Differentiating the Bias

$$C = \frac{1}{2m} \sum_{i=1}^{m} (Y - a^L)^2 = \frac{1}{2m} \sum_{i=1}^{m} (v)^2 \quad \text{where } v = Y - a^L$$

$$\frac{\partial c}{\partial b} = \frac{\partial c}{\partial v} \frac{\partial v}{\partial a^L} \frac{\partial a^L}{\partial b}$$

We know :

$$\frac{\partial a^L}{\partial b} = \begin{cases} 0 & \text{if } w^T x + b \leq 0 \\ 1 & \text{if } w^T x + b > 0 \end{cases}$$

$$\frac{\partial v}{\partial a^L} = \frac{\partial}{\partial a^L} (Y - a^L) = -1$$

$$\therefore \frac{\partial v}{\partial b} = \frac{\partial v}{\partial a^L} \cdot \frac{\partial a^L}{\partial b} = -1 \cdot \begin{cases} 0 & \text{if } w^T x + b \leq 0 \\ 1 & \text{if } w^T x + b > 0 \end{cases} = \begin{cases} 0 & \text{if} \ldots \\ -1 & \text{if} \ldots \end{cases}$$

We could find $\frac{\partial c}{\partial v}$ and multiply it by $\frac{\partial v}{\partial b}$, but it is easier to find $\frac{\partial c}{\partial b}$ and substitute derivatives along the way

$$\frac{\partial c}{\partial b} = \frac{\partial}{\partial b}\left(\frac{1}{2m}\sum_{i=1}^{m}(v^2)\right) = \frac{1}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial b}v^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\frac{\partial v^2}{\partial v}\frac{\partial v}{\partial b} = \frac{1}{2m}\sum_{i=1}^{m}2v\frac{\partial v}{\partial b}$$

$$= \frac{1}{m}\sum_{i=1}^{m}v\cdot\begin{cases}0 & \text{if } W^Tx+b\le 0\\ -1 & \text{if } W^Tx+b\end{cases} = \frac{1}{m}\sum_{i=1}^{m}\begin{cases}0 & \text{if}\ldots\\ -V & \text{if}\ldots\end{cases}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\begin{cases}0 & \text{if } W^Tx+b\le 0\\ -(y-a^L) & \text{if } W^Tx+b>0\end{cases}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\begin{cases}0 & \text{if } W^Tx+b\le 0\\ -(y-\max(0, W^Tx+b)) & \text{if } W^Tx+b>0\end{cases}$$

Piecewise makes $\max(0, W^Tx+b)$ redundant so we can remove

$$= \frac{1}{m}\sum_{i=1}^{m}\begin{cases}0 & \text{if } W^Tx+b\le 0\\ -(y-(W^Tx+b)) & \text{if } W^Tx+b>0\end{cases}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\begin{cases}0 & \text{if } W^Tx+b\le 0\\ W^Tx+b-y & \text{if } W^Tx+b>0\end{cases}$$

$$\frac{\partial c}{\partial b} = \frac{1}{m}\begin{cases}0 & \text{if } W^Tx+b\le 0\\ \sum_{i=1}^{m}(W^Tx+b-y) & \text{if } W^Tx+b>0\end{cases}$$

$\boxed{\text{This is a scalar}}$

3.7 Gradient Descent Intuition

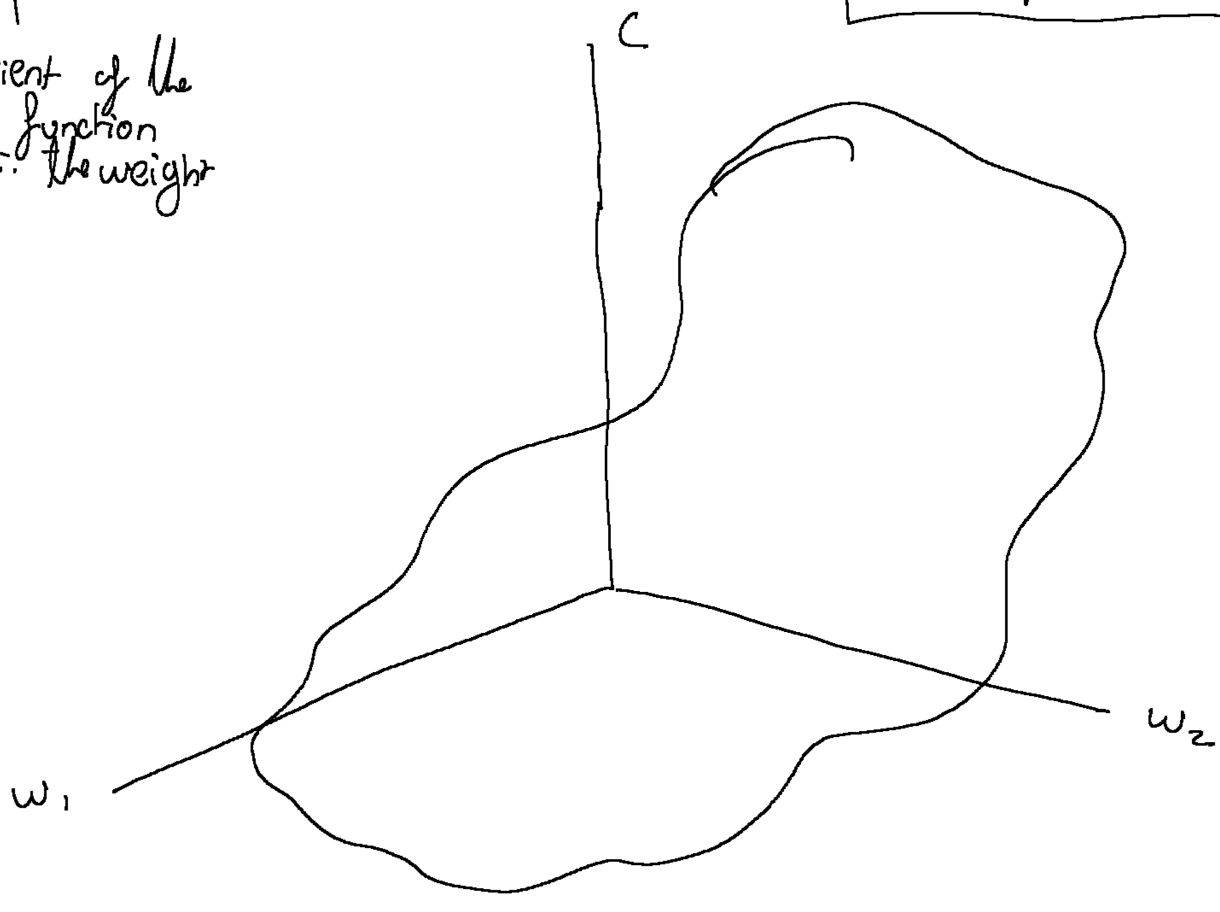
$$\frac{\partial c}{\partial \vec{w}} = \begin{bmatrix} e_i x_1 \\ e_i x_2 \\ \vdots \\ e_i x_n \end{bmatrix}$$

Lets take a network w/ 2 weights and graph it in 3-D

$$\nabla_w C = \begin{array}{cc} \begin{bmatrix} e_i x_1 \\ e_i x_2 \end{bmatrix} & \begin{array}{l} \rightarrow w_1 \\ \rightarrow w_2 \end{array} \end{array}$$

Gradient of the cost function w.r.t. the weight

> ★ Remember, the gradient of a function points in the direction of steepest ascent



We can then take the negative of the steepest gradient $(-\nabla_w C)$ as this will tell us how to most quickly decrease the cost

3.8 Gradient Descent Algorithm and SGD

$$\nabla_{w,b} C = \begin{bmatrix} \frac{\partial c}{\partial w_1} \\ \frac{\partial c}{\partial w_2} \\ \vdots \\ \frac{\partial c}{\partial w_n} \end{bmatrix}$$

We can "unravel" all the weights and biases into a vector that is the same length as $\nabla_{w,b} C$. We can call this vector Θ

Gradient descent is an iterative process. It takes many iterations to lower the cost by adjusting the weights and bias

$$\therefore \; \Theta = \Theta - \underbrace{\alpha}_{\text{Learning rate}} \nabla_{w,b} C = \begin{bmatrix} w_1 - \alpha \frac{\partial c}{\partial w_1} \\ w_2 - \alpha \frac{\partial c}{\partial w_2} \\ \vdots \\ w_n - \alpha \frac{\partial c}{\partial w_n} \\ b - \alpha \frac{\partial c}{\partial b} \end{bmatrix}$$

The learning rate plays a critical role in determining how big the "steps" of adjusting the weights and biases are

If the learning rate is too big, it could overshoot the minima and thus it may not converge

If the learning rate is too small, it could get "stuck" and not converge to the optimal point as it moves too slowly

Stochastic gradient descent (SGD): A modification of gradient descent where you calculate the gradient using just a small part of the observations instead of all. Can reduce computation time

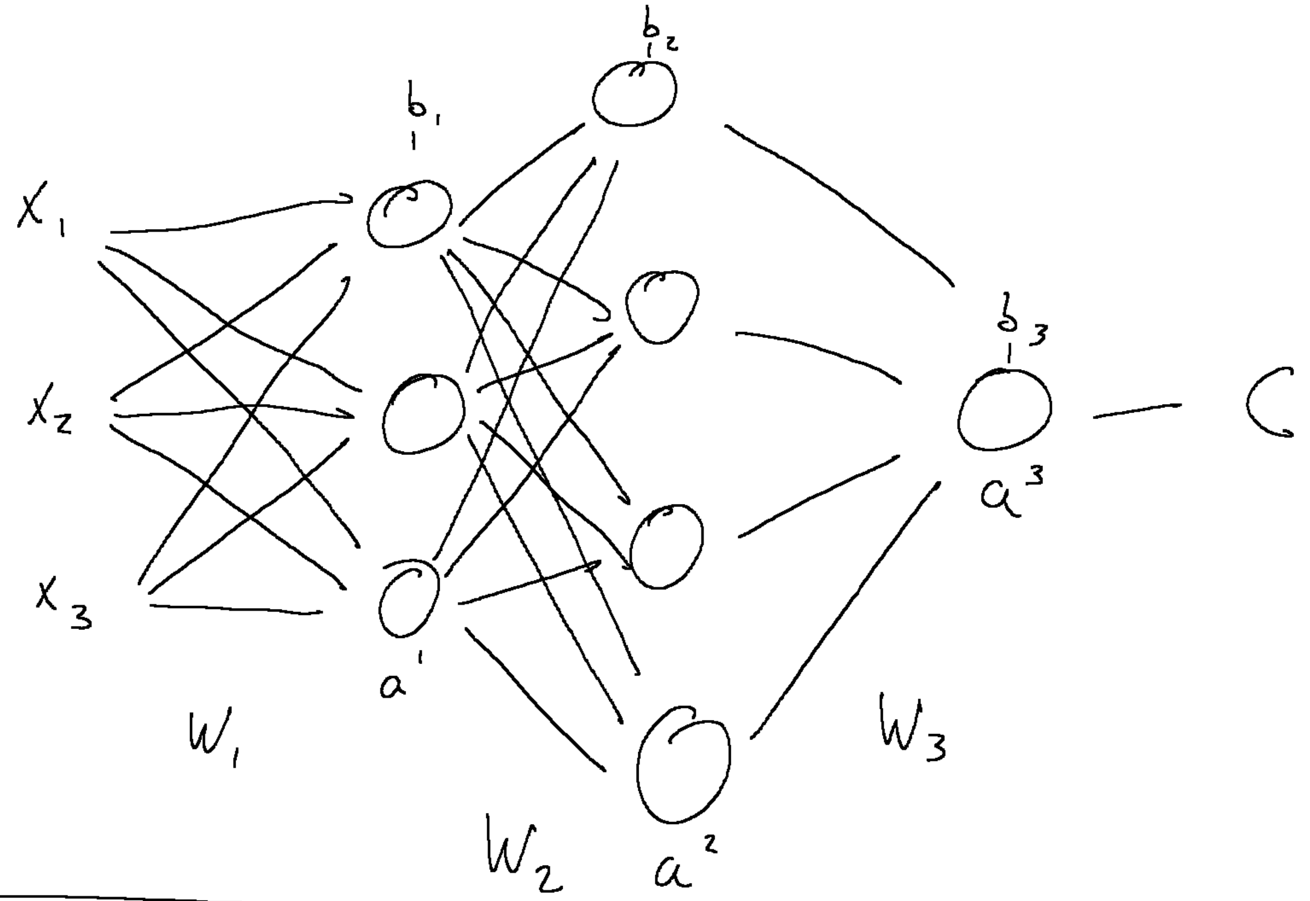
Batch stochastic gradient descent: The gradients are calculated and the decision variables are updated iteratively with a subset of observations, called minibatches

---

Lets say we have 1,000,000 training examples, m, and we take batches of size 120. Then we would get 8,333 batches (with the last one being partially full) all with 120 examples in the batch

We could then run gradient descent on each batch

3.9 Finding Derivatives of an Entire Layer (and why it doesn't work well)



If we want to see how changing w_1 affects the cost, we get:

$$\frac{\partial c}{\partial w_1} = \frac{\partial c}{\partial a^3} \frac{\partial a^3}{\partial a^2} \frac{\partial a^2}{\partial a^1} \frac{\partial a^1}{\partial w_1}$$

If we want to see how w_2 affects the cost:

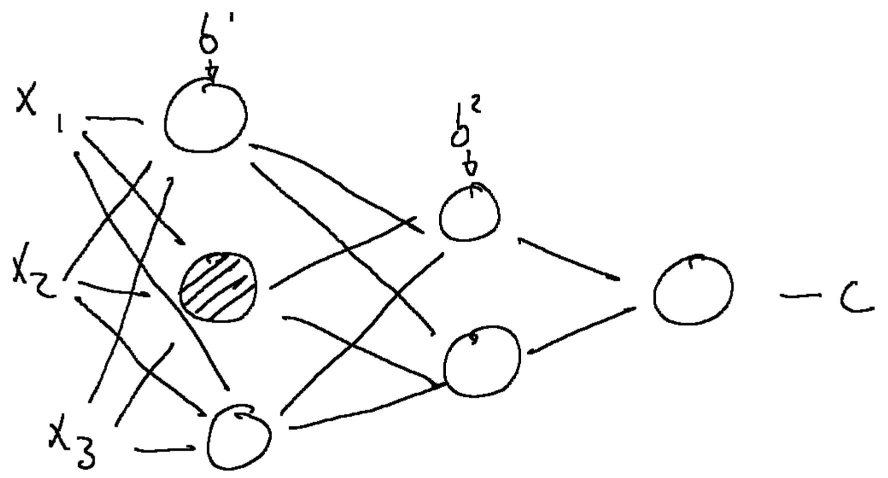
$$\frac{\partial c}{\partial w_2} = \frac{\partial c}{\partial a^3} \frac{\partial a^3}{\partial a^2} \frac{\partial a^2}{\partial w_2}$$

This is quite difficult to calculate everything "forward", so instead we use "back propogation" so that we don't have to recalculate the derivatives when we change a weight

PART IV

Backpropagation

4.1 The Error of a Node



We want to examine how the cost changes when we add a Value, Δ, to a node

$$a_2^1 = \sigma(\underbrace{W^T x + b + \Delta}_{z})$$

We would actually be adding the value to the z (before it goes through the activation function so it would be

$$z_2^1 + \Delta$$

Therefore we can say that the error of a node, δ, is given by

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

It shows us how much the cost will change by changing a value of a node

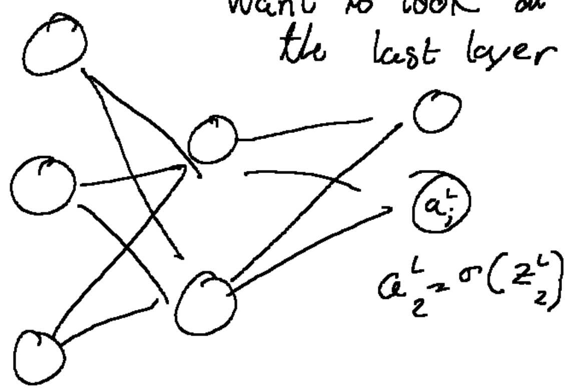
4.2 The Four Equations of Backpropogation

4.2.1 Equation 1: Finding the error of all nodes in the last layer

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

Error of j^{th} node in layer l

Want to look at a node in the last layer



Error for ONE node in the last layer:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L}$$

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \cdot \sigma'(z_j^L)$$

Derivative of activation function w.r.t. z_j^L

Error for EVERY node in the last layer:

$$\delta^L = \nabla_{a^L} C \cdot \sigma'(z^L) \quad \text{where} \quad \sigma'(z^L) = \begin{bmatrix} \sigma'(z_1^L) \\ \sigma'(z_2^L) \\ \vdots \\ \sigma'(z_n^L) \end{bmatrix}$$

This will be a vector of errors for each node in the last layer

4.2.2 Equation 2: Finding the error of any node (δ^l)

If we know the errors of the nodes in layer $l+1$, we can use this to calculate the errors in node l:

$$\delta^l = \left((W^{l+1})^T \delta^{l+1} \right) \otimes \sigma'(z^l)$$

weights matrix of layer $l+1$

vector of errors of layer $l+1$

elementwise Product

Derivative of the activations of the z's in layer l

Lets take an example of a two layer network

First layer — second layer

$$\sigma(W_1 a_0 + b_1) + \sigma(W_2 a_1 + b_2)$$

Lets try to find the error of the first layer $\left(\frac{\partial C}{\partial z_1}\right)$

$$\underbrace{\frac{\partial C}{\partial z_1}}_{\delta^1} = \underbrace{\frac{\partial C}{\partial z_2}}_{\delta^2} \; \underbrace{\frac{\partial z_2}{\partial a_1}}_{(W_2)^T} \; \underbrace{\frac{\partial a_1}{\partial z_1}}_{\sigma'(z_1)}$$

$\therefore$

$$\delta^1 = \left((W_2)^T \cdot \delta^2 \right) \otimes \sigma'(z_1)$$

4.2.3. Equation 3: Finding the derivative of the cost w.r.t. any bias

$$\frac{\partial c}{\partial b^l} = \delta^l$$

Lets take an example of a two layer network

First layer Second layer

$$\sigma(W_1 a_0 + b_1) \Rightarrow \sigma(W_2 a_1 + b_2)$$

Lets say we want to find $\frac{\partial c}{\partial b_1}$

We already have the error of layer 1 $\left(\delta^1 = \frac{\partial c}{\partial z_1}\right)$

$$\frac{\partial c}{\partial b_1} = \underbrace{\frac{\partial c}{\partial z_1}}_{\delta_1} \underbrace{\frac{\partial z_1}{\partial b_1}}_{1}$$

$\therefore$

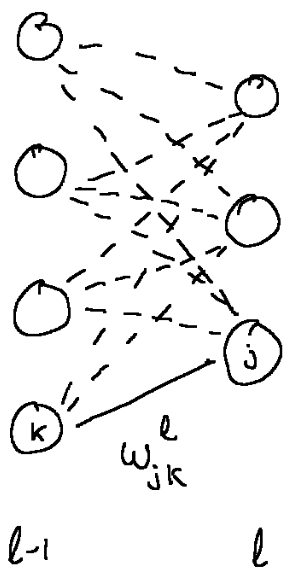
$$\frac{\partial c}{\partial b_1} = \delta_1$$

4.2.4 Equation 4: Finding the derivative of the cost w.r.t. any weight

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \cdot \delta^l_j$$

This is a scalar not a vector we are finding the derivative of a specific weight going into the j^{th} node in layer l coming from the k^{th} node in layer $l-1$

Lets take an example network:



Trying to find $\frac{\partial C}{w^l_{jk}}$

We have $\delta^l_j = \frac{\partial C}{\partial z^l_j}$

$$\frac{\partial C}{\partial w^l_{jk}} = \underbrace{\frac{\partial C}{\partial z^l_j}}_{\delta^l_j} \underbrace{\frac{\partial z^l_j}{\partial w^l_{jk}}}_{a^{l-1}_k}$$

$\therefore$

$$\frac{\partial C}{\partial w^l_{jk}} = \delta^l_j \cdot a^{l-1}_k$$

4.2.5 Equation 4: Finding the derivative of the cost w.r.t. any weight

VECTORIZED

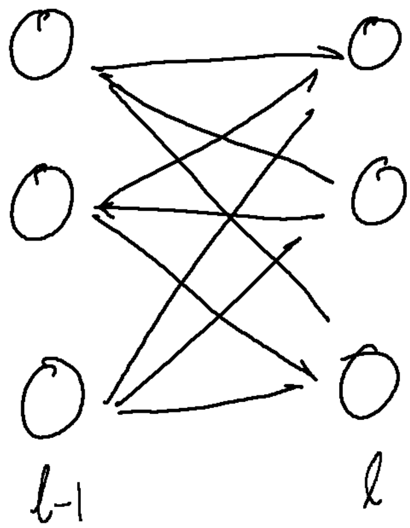
We know $\dfrac{\partial C}{\partial W^{l}_{jk}} = a^{l-1}_{k}\,\delta^{l}_{j}$

We now want to find $\dfrac{\partial C}{\partial W^{l}}$ (for the entire weight matrix)

Remember,

$$W^{l} = \begin{bmatrix} W_{11} & W_{12} & \dots & W_{1k} \\ W_{21} & W_{22} & \dots & W_{2k} \\ \vdots & \vdots & & \vdots \\ W_{j1} & W_{j2} & \dots & W_{jk} \end{bmatrix}$$

So for this network, the weights matrix would be



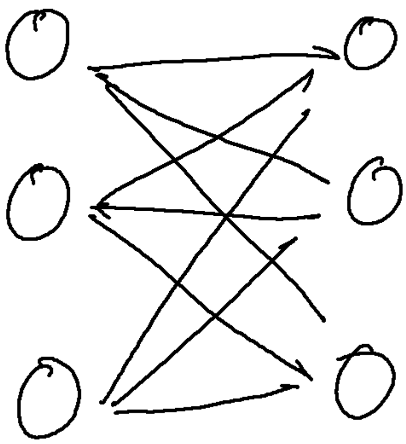
$$W^{l} = \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{bmatrix}$$

Therefore the derivative of the matrix would be the same size (3,3)

$$\frac{\partial c}{\partial w^{l}} = \begin{bmatrix} \frac{\partial c}{\partial w_{11}} & \frac{\partial c}{\partial w_{12}} & \frac{\partial c}{\partial w_{13}} \\ \frac{\partial c}{\partial w_{21}} & \frac{\partial c}{\partial_{22}} & \frac{\partial c}{\partial w_{23}} \\ \frac{\partial c}{\partial w_{31}} & \frac{\partial c}{\partial w_{32}} & \frac{\partial c}{\partial w_{33}} \end{bmatrix}$$

$$\boxed{\frac{\partial c}{\partial w^l_{jk}} = a^{l-1}_k \, \delta^l_j}$$

$$W_2 = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$



$$W_2 = \begin{bmatrix} a^1_1 \cdot \delta^2_1 & a^1_2 \cdot \delta^2_1 & a^1_3 \cdot \delta^2_1 \\ a^1_1 \cdot \delta^2_2 & a^1_2 \cdot \delta^2_2 & a^1_3 \cdot \delta^2_2 \\ a^1_1 \cdot \delta^2_3 & a^1_2 \cdot \delta^2_3 & a^1_3 \cdot \delta^2_3 \end{bmatrix}$$

Lets take a vector of all the errors of layer 2 and the activations from layer 1

$$\vec{\delta}^2 = \begin{bmatrix} \delta^2_1 \\ \delta^2_2 \\ \delta^2_3 \end{bmatrix} \qquad \vec{a}^1 = \begin{bmatrix} a^1_1 \\ a^1_2 \\ a^1_3 \end{bmatrix}$$

Therefore we get:

$$\frac{\partial c}{\partial W^l} = \vec{\delta}^l \cdot \left(\overrightarrow{a^{l-1}}\right)^T$$

4.3 Tying part III and part IV together

$$\frac{\partial c}{\partial W^{\ell}} = \vec{\delta}^{\ell} \left(\overrightarrow{a^{\ell-1}}\right)^T \qquad \frac{\partial c}{\partial b} = \delta^{\ell}$$

$$\frac{\partial c}{\partial W} = \begin{cases} \vec{0} & \text{if } W^T x + b \leq 0 \\ \frac{1}{m} \sum_{i=1}^{m} \underbrace{(W^T x + b - y)}_{e^i} x^T = \frac{1}{m} \sum_{i=1}^{m} e_i x^T & \text{if } W^T x + b > 0 \end{cases}$$

$$\frac{\partial c}{\partial b} = \begin{cases} 0 & \text{if } W^T x + b \leq 0 \\ \frac{1}{m} \sum_{i=1}^{m} \underbrace{(W^T x + b - y)}_{e^i} = \frac{1}{m} \sum_{i=1}^{m} e_i & \text{if } W^T x + b > 0 \end{cases}$$

Taking one training example where $W^T x + b > 0$, we get

$$\frac{\partial c}{\partial W} = e_i x^T \qquad \frac{\partial c}{\partial b} = e_i$$

These are the same as the backprop equations, but just for one node

4.4 The Backpropagation Algorithm

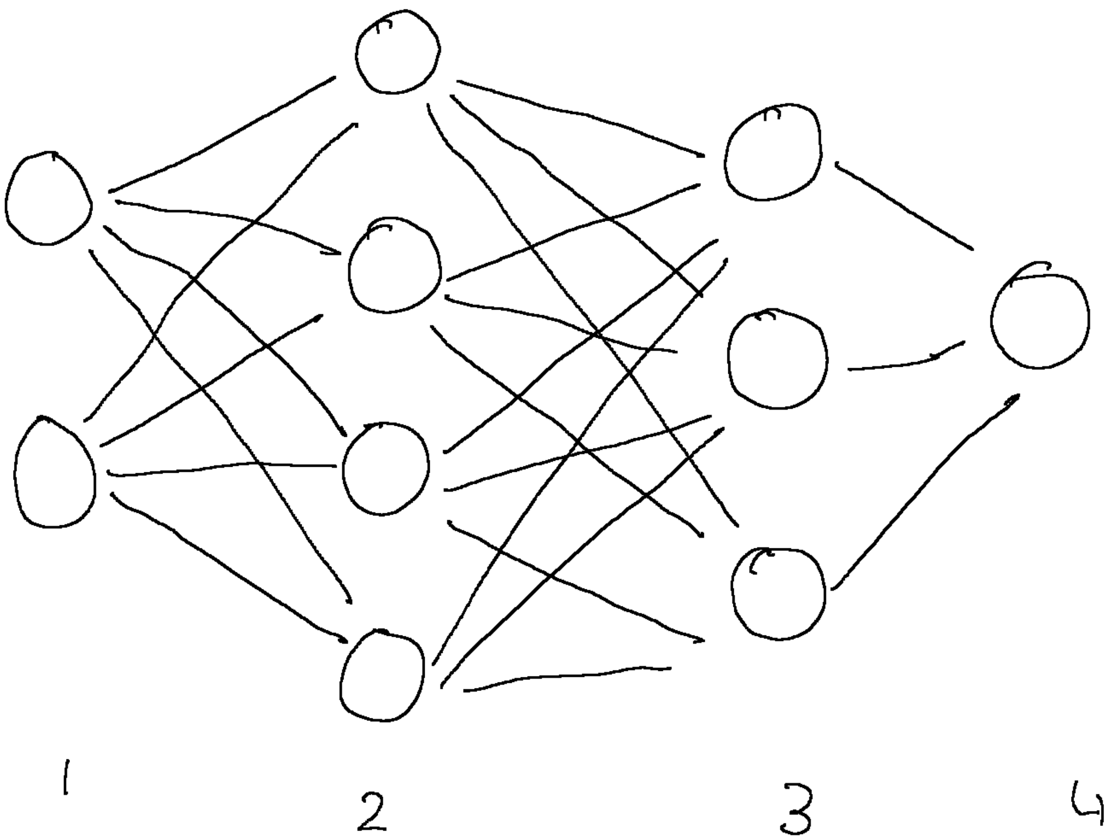
Backprop Algorithms:

Eq 1: $$\delta^L = \nabla_a C \otimes \sigma'(z^L)$$

Eq 2: $$\delta^l = \left(\left(W^{l+1}\right)^T \delta^{l+1}\right) \otimes \sigma'(z^l)$$

Eq 3: $$\frac{\partial C}{\partial b} = \delta^l$$

Eq 4: $$\frac{\partial C}{\partial W^l} = \delta^l \left(a^{l-1}\right)^T$$



The Algorithm:

1. Forward propogate through the network
   ↳ Calculate and store the z^l values and a^l values

2. Compute the cost

3. Back propogate through the network to calculate all the errors
   ↳ 3.1 Use Eq. 1 to calculate errors of the last layer (δ^L)

   ↳ 3.2 use Eq. 2 to calculate the errors of each network (δ^l) using the errors calculated for the layer after (δ^{l+1})

4. Calculate the derivatives of the cost w.r.t the bias and weights
   ↳ 4.1 Use the errors calculated (δ^l) and Eq. 3 to find $\frac{\partial C}{\partial b}$

   ↳ 4.2 Use the errors calculated (δ^l), the activations (a^l) and Eq 4 to calculate $\frac{\partial C}{\partial W^l}$

5. Using everything we calculated, we can find $\nabla_{w,b} C$ and use gradient descent to update the weights and biases

$$\Theta = \Theta - \alpha \nabla_{w,b} C$$