Let $W^{(1)} = \begin{bmatrix} -2 & 4 & -1 \\ 6 & 0 & -3 \end{bmatrix}$ & $\vec{b^{(1)}} = \begin{bmatrix} 0.1 \\ -2.5 \end{bmatrix}$

in $\vec{a^{(1)}} = \sigma(W^{(1)} \cdot \vec{a^{(0)}} + \vec{b^{(1)}})$ with

& $\vec{a^{(0)}} = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.1 \end{bmatrix}$

Find $\vec{a^{(1)}}$.

Soln :- $W^{(1)} \cdot \vec{a^{(0)}} = \begin{bmatrix} -2 & 4 & -1 \\ 6 & 0 & -3 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.4 \\ 0.1 \end{bmatrix} = \begin{bmatrix} -0.6+1.6-0.1 \\ 1.8-0.3 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 1.5 \end{bmatrix}$

$\therefore W^{(1)} \cdot \vec{a^{(0)}} + \vec{b^{(1)}} = \begin{bmatrix} 1.0 \\ -1.0 \end{bmatrix}$

$\Rightarrow \vec{a^{(1)}} = \begin{bmatrix} \sigma(1.0) \\ \sigma(-1.0) \end{bmatrix} = \begin{bmatrix} 0.76 \\ -0.76 \end{bmatrix}$.

More simple neural networks

AIM : Iteratively update the values of all weights & Biases so that the network learns to perform tasks (like classify data) based on a set of training examples.

Training a network means finding the right weights & biases using the training example. (which are pairs of matching inputs & outputs)

Backpropogation is a classical training method. Here, we look ~~at the~~ first at the output layer & then calculate gradients (of the cost. func) backward towards the input layer with chain rule.

In a many-layered neural network, we have

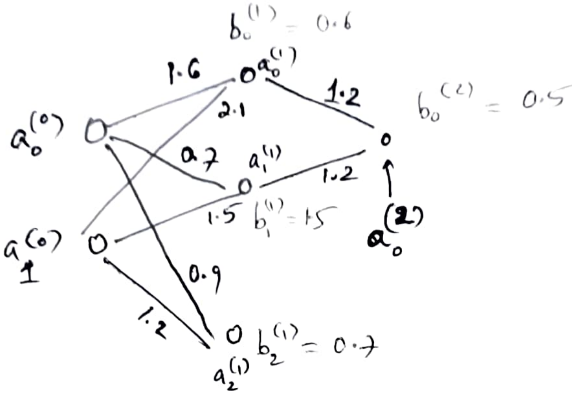
$$a^{(L)} = \sigma(W^{(L)} \cdot a^{(L-1)} + b^{(L)})$$

when L denotes the final layer.

Let us assume the case where there is one training example, say input is X & output is Y. X will be a vector of same size as $a^{(0)}$ & Y will be of same size as $a^{(L)}$.

~~$a^{(L)}$ = predicted~~ For neural network to start prediction, we will randomly initialise all the weights & biases.

For eg.



Now, for these values of weights & biases, I can find that the error (cost) incurred is

$$C = \frac{1}{2}(a_0^{(2)} - Y)^2$$

For the same training example, if I think of weights & biases as variables, then cost will be a function which depends on the variable $a_0^{(2)}$ which now depends on these weights & biases.

$$C(a_0^{(2)}) = \frac{1}{2}(a_0^{(2)} - Y)^2$$

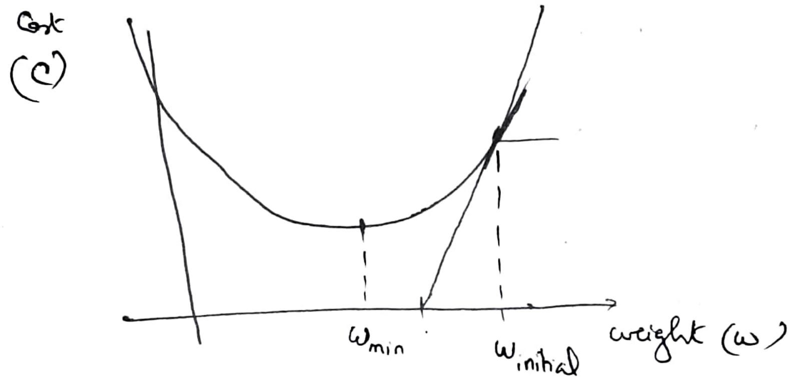
To MINIMIZE COST

~~To minimise cost~~ Suppose for the initial weights & biases, the cost value was too high, we update the weights & biases (in an iterative manner) so as to minimise the cost.

Note that this updation will change $a_0^{(2)}$ & in turn minimise C.

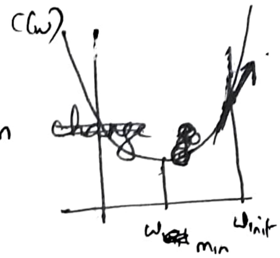
Consider a simple example $a^{(0)} \xrightarrow{w} a^{(1)}$ where $\sigma(x) = x$ & $b^{(1)} = 0$, ie, $a^{(1)} = w a^{(0)} \Rightarrow C = \frac{1}{2}(a^{(1)} - Y)^2 = (wa^{(0)} - Y)^2$

(Parabola)


Cost
(C)
w_{min}
$w_{initial}$
weight (w)

If weight is too large or too small, (cost) error is high & ~~hence to~~

at w_{min}, it has minimum cost.

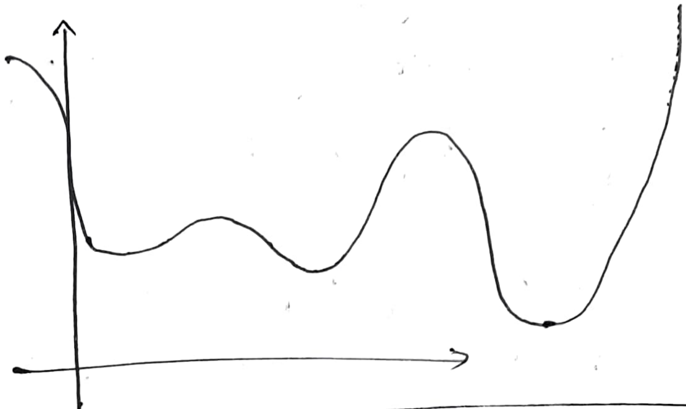


If at your randomly chosen w_{init}, $\frac{\partial C}{\partial w}$ is +ve, then ~~change~~ reduce C by going in opposite dirn. of gradient.

However,

A Brief Look at Challenges :-

→ Multiple local minimas



In class, I had shown a challenge which appears in another iterative approach called Newton's Method. Here, we will use only Gradient Descent Method.

→ Gradient exploding & vanishing problems ~~exploding~~

Chain Rule ~~Backpropagation~~ in Case 1

$$a^{(0)} \circ \xrightarrow{\ w\ } \circ\, a^{(1)}$$

Use $w^{(1)} = w$ for (simplicity).

Recall $a^{(1)} = \sigma(w^{(1)} a^{(0)} + b^{(1)}) = \sigma(w a^{(0)} + b)$

Here, $C = (a^{(1)} - y)^2$,

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}}$$

Implementation of gradient can be made less expensive by using intermediate variables, say

$$z^{(1)} = w^{(1)} a^{(0)} + b^{(1)} \quad \text{or} \quad z = w a^{(0)} + b$$

~~ie, $\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a^{(1)}} \times \frac{\partial a^{(1)}}{\partial w} \times \frac{\partial w}{\partial}$~~

Then, $a^{(1)} = \sigma(z^{(1)})$ & hence,

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z} \frac{\partial z}{\partial w}$$

~~Eg 1) Consider the neural network~~

~~$a^{(0)} \xrightarrow{\;w^{(1)}\;} a^{(1)}$~~

~~Let $\sigma(x) = x$ (linear activation ~~action~~ func.) & $b^{(1)} = 0$~~

<u>Forward propagation</u>

The process of obtaining the ~~value~~ vector $a^{(1)}$ corresponding to a given (random/updated) set of Biases & Weight matrices by considering a specific training example & thus, finding the error value is called <u>forward propagation</u>. (FP)

The process of propagating the error (obtained at final layer) ~~& passing~~ ~~&~~ to previous layers so as to update the weights & bias (in a ~~required~~ desired manner) is called <u>backpropagation</u>. (BP)

The process of FP & BP are repeated until the value of error fn is 'satisfactory'.

Backpropagation basics

Recall that $a^{(n)} = \sigma(z^{(n)})$ is the feedforward equation at the n^{th} layer where $z^{(n)} = W^{(n)} a^{(n-1)} + b^{(n)}$

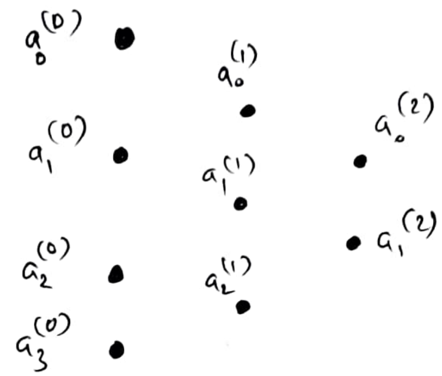
Cost $C = \frac{1}{2}\|a^{(n)} - y\|^2$ $\qquad 0 \le n \le N$

(For a single training example)

In order to update W & b, we need to determine $\overset{\frac{\partial C}{\partial W}}{\overset{\shortparallel}{\nabla_W C}}$ & $\overset{\frac{\partial C}{\partial b}}{\overset{\shortparallel}{\nabla_b C}}$ (& other relevant gradients intermittent)

(I will also write $\frac{\partial C}{\partial W}$ & $\frac{\partial C}{\partial b}$ from now on)

Consider an example



Here, $\cancel{a^{(2)} = W^{(2)} a^{(1)} + b^{(2)}}$

$$C = \frac{1}{2}\|a^{(2)} - y\|^2 \quad \text{where}$$

$$a^{(2)} = \sigma(z^{(2)}) \quad \& \quad z^{(2)} = W^{(2)} a^{(1)} + b^{(2)}$$

$$a^{(1)} = \sigma(z^{(1)}) \quad \& \quad z^{(1)} = W^{(1)} a^{(0)} + b^{(1)}$$

$$\frac{\partial C}{\partial W^{(2)}} = \frac{\partial C}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial W^{(2)}} \quad — (1) \quad \&$$

$$\frac{\partial C}{\partial W^{(1)}} = \frac{\partial C}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial W^{(1)}} \quad — (2)$$

Here, if $\frac{\partial C}{\partial W^{(2)}}$ has been computed, then few of those ~~most of the~~ computations in (1) can be reused to compute $\frac{\partial C}{\partial W^{(1)}}$. Further,

$$(2) \Leftrightarrow \quad \frac{\partial C}{\partial W^{(1)}} = \frac{\partial C}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial W^{(1)}}$$

(2) can be generalised as (for N layered network)

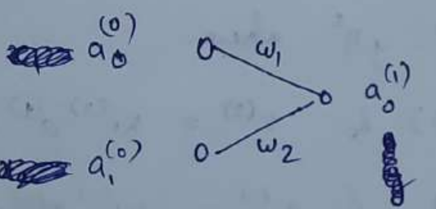
$$\frac{\partial C}{\partial w^{(i)}} = \overbrace{\frac{\partial C}{\partial a^{(N)}}\frac{\partial a^{(N)}}{\partial a^{(N-1)}}\frac{\partial a^{(N-1)}}{\partial a^{(N-2)}}\cdots\frac{\partial a^{(i+1)}}{\partial a^{(i)}}}\frac{\partial a^{(i)}}{\partial z^{(i)}}\frac{\partial z^{(i)}}{\partial w^{(i)}}$$

N^{th} layer to i^{th} layer

Illy,
$$\frac{\partial C}{\partial w^{(i-1)}} = \underbrace{\frac{\partial C}{\partial a^{(N)}}\frac{\partial a^{(N)}}{\partial a^{(N-1)}}\frac{\partial a^{(N-1)}}{\partial a^{(N-2)}}\cdots\frac{\partial a^{(i+1)}}{\partial a^{(i)}}}_{\text{Reusable computations from the } i^{th} \text{ layer}}\frac{\partial a^{(i)}}{\partial a^{(i-1)}}\frac{\partial a^{(i-1)}}{\partial z^{(i-1)}}\frac{\partial z^{(i-1)}}{\partial w^{(i-1)}}$$

Now, let us do a problem involving neural networks with ~~be~~ forward & back propagation, ~~with~~ for single training example.

Pr 1) Consider ~~a~~ the following neural network



$b^{(1)}$ – bias

Using the activation function, $\sigma(x) = \frac{1}{1+e^{-x}}$ (sigmoid) & learning rate $\alpha = 0.1$, perform 1 step of FP & BP for the training example

$X = (x_1, x_2)$	y
$(0.8, 0.6)$	1.0

with initial weights & biases as given here:-
$w_1 = 0.3$, $w_2 = 0.5$
$b = 0.1$

Soln:- $\sigma'(x) = -(1+e^{-x})^{-2} \times (-e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2}$.

$$a^{(0)} = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} ; W^{(1)} = \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}^T ; b^{(1)} = 0.1 ; a^{(1)} = \sigma($$

$a^{(1)} = \sigma(z^{(1)})$ where $z^{(1)} = W^{(1)}a^{(0)} + b^{(1)} = w_1 a_0^{(0)} + w_2 a_1^{(0)} + b^{(1)}$

~~Let us use~~ <u>Forward propagation</u>

$$z^{(1)} = \begin{bmatrix} 0.3 & 0.5 \end{bmatrix} \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} + 0.1 = 0.64$$

$$a^{(1)} = \sigma(z^{(1)}) = 0.6547$$

$$C = \frac{1}{2}(a^{(1)} - y)^2 = \frac{1}{2}[0.6547 - 1]^2 = \frac{0.1192}{2} = 0.0596$$

<u>Backpropagation</u>

$$w_1^{new} = w_1^{old} - \alpha \frac{\partial C}{\partial w_1}(W^{(1),old})$$

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial a^{(1)}} \times \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial w_1}$$

$$= \frac{1}{2} \times 2 \times (a^{(1)} - y) \times \sigma'(z^{(1)}) \times \frac{\partial z^{(1)}}{\partial w_1}$$

$$z^{(1)} = w_1 a_0^{(0)} + w_2 a_1^{(0)} + b^{(1)}$$

$$\Rightarrow \frac{\partial z^{(1)}}{\partial w_1} = a_0^{(0)} = 0.8 \quad \& \quad \frac{\partial z^{(1)}}{\partial w_2} = a_1^{(0)} = 0.6$$

$$\text{III}^{ly}, \frac{\partial C}{\partial l}$$

$$W^{(1),new} = W^{(1),old} - 0.1 \times \begin{bmatrix} \frac{\partial C}{\partial w_1}(W^{(1)old}) \\ \frac{\partial C}{\partial w_2}(W^{(1)old}) \end{bmatrix}^T$$

Only diff between these two is this term

Let us simply $W^{(1),new}$ & $W^{(1),old}$ notations to simply W^{new} & W^{old}.

$$\left.\frac{\partial C}{\partial w_1}\right|_{W^{old}} = (0.6547 - 1) \times \frac{e^{-0.64}}{(1+e^{-0.64})^2} \times 0.8$$

$$= -0.3453 \times 0.226 \times 0.8 = -0.0624$$

$$\frac{\partial C}{\partial w_2} = \frac{\partial C}{\partial a^{(1)}} \times \frac{\partial a^{(1)}}{\partial z^{(1)}} \times \frac{\partial z^{(1)}}{\partial w_2}$$

$$\Rightarrow \left.\frac{\partial C}{\partial w_2}\right|_{W^{old}} = -0.3453 \times 0.226 \times 0.6 = -0.047$$

$$\therefore W^{new} = \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}^T - 0.1 \begin{bmatrix} -0.0624 \\ -0.047 \end{bmatrix}^T$$

$$= \begin{bmatrix} 0.30624 \\ 0.5047 \end{bmatrix}^T$$

We can form the foll. table

$W^{(1)}$	$a^{(0)}$	$\frac{\partial C}{\partial W}$	$\frac{\partial C}{\partial W^{(1)}}$	$\cancel{\frac{\partial C}{\partial w_2}}$	$W^{(1),new}$	$\frac{\partial C}{\partial b^{(1)}}$	$B^{(1),new}$
$\begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}^T$	$\begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}$	0.0596	$\begin{bmatrix} -0.0624 \\ -0.047 \end{bmatrix}^T$	~~-0.047~~	$\begin{bmatrix} 0.30624 \\ 0.5047 \end{bmatrix}$	-0.078	~~0.054~~ 0.1078

Here, $\frac{\partial C}{\partial b^{(1)}} = \frac{\partial C}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial b^{(1)}}$

$$\Rightarrow \left.\frac{\partial C}{\partial b^{(1)}}\right|_{W^{old}} = -0.3453 \times 0.226 \times 1 = -0.078$$

& hence, $b^{new} = 0.1 - 0.1 \times (-0.078) = 0.1078$

Thus, we form the foll table (2nd step exercise)

$b^{(1)}$	$W^{(1)}$	$a^{(1)}$	C	$\nabla_{W^{(1)}} C$	$\frac{\partial C}{\partial b^{(1)}}$	$W^{(1),new}$	$b^{(1),new}$
0.1	$\begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}^T$	0.655	0.0596	$\begin{bmatrix} -0.0624 \\ -0.047 \end{bmatrix}^T$	-0.078	$\begin{bmatrix} 0.30624 \\ 0.5047 \end{bmatrix}^T$	0.1078
0.1078	$\begin{bmatrix} 0.30624 \\ 0.5047 \end{bmatrix}^T$	0.6583	0.0584	$\begin{bmatrix} -0.0615 \\ -0.0461 \end{bmatrix}^T$	-0.0768	$\begin{bmatrix} 0.3124 \\ 0.5093 \end{bmatrix}^T$	0.1155

Linear Regression problems : Background & brief overview

Let input $X = (x_1, x_2, \ldots, x_n)$ & $Y \in \mathbb{R}$. Let there be m training examples

Training example	x_1	x_2	x_3		x_n	Y
1	x_1^1	x_2^1	x_3^1	$\ldots$	x_n^1	y^1
2	x_1^2	x_2^2	x_3^2,	$\ldots$	x_n^2	
$\vdots$						
m	x_1^m	x_2^m	x_3^m	$\ldots$	x_n^m	y^m

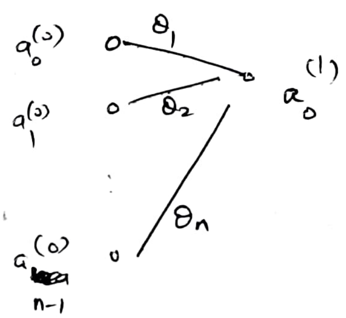
~~Let we want to~~ Let us say $h_\theta(X) = \sum_{i=0}^{n} \theta_i x_i$

$\theta_0 \rightarrow$ bias

$\left.\begin{matrix}\theta_1 \\ \vdots \\ \theta_n\end{matrix}\right\}$ weights

$\theta_1, \ldots, \theta_n$ will be the weights of a neural network with n inputs & 1 output & θ_0 will be its bias.

Let $\theta = \begin{bmatrix}\theta_1 \\ \theta_2 \\ \vdots \\ \theta_n\end{bmatrix}$



Cost fn $\quad J(\theta) = \frac{1}{2} \sum_{i=0}^{m} \left[h_\theta(x^i) - y^{(i)}\right]^2$

To minimise $J(\theta)$, we ~~choose~~ start with an initial guess of θ & use the search algorithm 'Gradient Descent' that repeatedly changes θ to make $J(\theta)$ smaller & smaller according to

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

This update is performed on all $0 \leq j \leq n$ simultaneously.

α - learning rate

Since θ is added with a vector along the $-$ve of the gradient of J, ~~then~~ the new θ obtained will be reducing $J(\theta)$

$$\begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_n\end{bmatrix}^{new} = \begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_n\end{bmatrix}^{old} - \alpha \begin{bmatrix}\frac{\partial J}{\partial \theta_0}(\theta^{old})\\ \frac{\partial J}{\partial \theta_1}(\theta^{old})\\ \vdots\\ \frac{\partial J}{\partial \theta_n}(\theta^{old})\end{bmatrix}$$

Computing $\frac{\partial J}{\partial \theta}$

$$J = \frac{1}{2}\sum_{i=1}^{m}\left(\theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_n x_n^{(i)} - y\right)^2$$

$$= \frac{1}{2}\sum_{i=1}^{m}\left(\theta_j x_j^{(i)} + \left(\sum_{\substack{k=0\\k\neq j}}^{n}\theta_k x_k^{(i)}\right) - y\right)^2$$

$$\therefore \frac{\partial J}{\partial \theta_j}(\theta) = \frac{1}{2}\sum_{i=1}^{m}\frac{\partial}{\partial \theta_j}\left[\theta_j x_j^{(i)} + \left(\sum_{\substack{k=0\\k\neq j}}^{n}\theta_k x_k^{(i)}\right) - y\right]^2$$

$$= \frac{1}{2}\sum_{i=1}^{m} 2\left(\theta_j x_j^{(i)} + \left(\sum_{\substack{k=0\\k\neq j}}^{n}\theta_k x_k^{(i)}\right) - y\right) x_j^{(i)}$$

$$= \sum_{i=1}^{m} x_j^{(i)}\left(h_\theta(x^{(i)}) - y\right)$$

For ~~a single~~ a single training eg (ie, $m=1$), say $\boxed{\left(\left(x_1^{(1)},\dots,x_n^{(1)}\right), y^{(1)}\right)}$

$$\frac{\partial J}{\partial \theta_j}(\theta) = x_j\left(h_\theta(x^{(1)}) - y^{(1)}\right)$$

Algorithm in batch gradient descent

Repeat until convergence,

$$\begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_n\end{bmatrix} = \begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_n\end{bmatrix} - \alpha\begin{bmatrix}\sum_{i=1}^{m} x_0^{(i)}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\\ \vdots\\ \sum_{i=1}^{m} x_n^{(i)}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\end{bmatrix}$$

Stochastic gradient descent

Here, we don't subtract the whole $\frac{\partial J}{\partial \theta_j}$ from θ_j, but a part, namely that corr. to each training example one at a time

algorithm

For $i = 1$ to m

$$\{ \quad \theta_j = \theta_j - \alpha x_j^{(i)} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \quad \}$$

First training example's component from $\frac{\partial J}{\partial \theta_0}$

ie,

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \longrightarrow \begin{bmatrix} \theta_0 - \alpha x_j^{(1)} \left(h_\theta(x^{(1)}) - y^{(1)} \right) \\ \vdots \\ \theta_n - \alpha x_j^{(1)} \left(h_\theta(x^{(1)}) - y^{(1)} \right) \end{bmatrix}$$

$\boxed{\text{At } i = 1}$

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \longrightarrow \begin{bmatrix} \theta_0 - \alpha x_0^{(1)} \left(h_\theta(x^{(1)}) - y^{(1)} \right) \\ \theta_1 - \alpha x_1^{(1)} \left(h_\theta(x^{(1)}) - y^{(1)} \right) \\ \vdots \\ \theta_n - \alpha x_1^{(1)} \left(h_\theta(x^{(1)}) - y^{(1)} \right) \end{bmatrix} = \theta^{new}$$

$$\downarrow$$

$\boxed{\text{At } i = 2}$

$$\begin{bmatrix} \theta_0^{new} - \alpha x_0^{(2)} \left(h_{\theta^{new}}(x^{(2)}) - y^{(2)} \right) \\ \theta_1^{new} - \alpha x_0^{(2)} \left(h_{\theta^{new}}(x^{(2)}) - y^{(2)} \right) \\ \vdots \\ \theta_n^{new} - \alpha x_0^{(2)} \left(h_{\theta^{new}}(x^{(2)}) - y^{(2)} \right) \end{bmatrix}$$

$$\downarrow$$

and so on

When $m = 1$, stochastic G.D = batch G.D

Eg 1) Consider the neural network

$$a^{(0)} \;\overset{w}{\rule{3em}{0.4pt}}\; a^{(1)}$$

Let the activation function be linear, i.e., $\sigma(x) = x$ & $b^{(1)} = 0$.

Initialise $w = 0.8$ & perform 3 steps of FP & BP for the training example $\boxed{x = 1.5 \,|\, y = 0.5}$

Soln :- $a^{(1)} = \sigma(a^{(0)} w^{(1)} + b^{(1)}) = 0.8\, a^{(0)}$

$$C = (a^{(1)} - y)^2$$

1st step of FP

Putting $a^{(0)} = 1.5$, we get $a^{(1)} = 0.8 \times 1.5 = 1.2$

Then, $C = (1.2 - 0.5)^2 = 0.49$.

1 step of BP

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a^{(1)}} \times \frac{\partial a^{(1)}}{\partial w} = 2(a^{(1)} - y) \times \frac{\partial a^{(1)}}{\partial w}$$

Since we want to update w, write $a^{(1)} = a^{(0)} w = 1.5 w$

~~1st step of BP~~

We now have the entries of the 1st row of the following table

$a^{(0)}$	w_{old}	b	$a^{(1)}$	Cost	$\left.\frac{\partial C}{\partial w}\right\|_{w=w_{old}}$ ~~$= 3(1.5w_{old} - 0.5)$~~	$w_{new} = w_{old} - r\frac{\partial C}{\partial w}(w_{old})$.
1.5	0.8	0	1.2	0.49		

To determine $\frac{\partial C}{\partial w}(w_{old})$, note that $\frac{\partial C}{\partial a^{(1)}} = 2(a^{(1)} - y)$ & $\frac{\partial a^{(1)}}{\partial w} = a^{(0)}$

$$\therefore \frac{\partial C}{\partial w}(0.8) = 2(1.2 - 0.5) \times 1.5 = 2.1$$

$$r = 0.1 \Rightarrow w_{new} = w_{old} - r\frac{\partial C}{\partial w}(0.8)$$

$$= 0.8 - 0.1 \times 2.1 = 0.59$$

This completes the entries of 1st row. Thus, we have performed 1 round of FP & BP, & adjusted weights. Now, let us see the change in the cost due to change in w. This leads to 2nd step of FP

2nd step of FP

$a^{(1)} = w\,a^{(0)}$ & $C = (a^{(1)} - y)^2$ where $w = 0.59$ (instead of 0.8)

So, $w_{old} = 0.59$.

$\Rightarrow a^{(1)} = 0.59 \times 1.5 = 0.885$ & $C = 0.148225$

$$\text{Now, } \frac{\partial C}{\partial w}(w_{old}) = \frac{\partial C}{\partial a^{(1)}} \cdot \frac{\partial a^{(1)}}{\partial w} = 2(a^{(1)} - 0.5) \times 1.5 = 1.155$$

~~$\Rightarrow w_{new} = 0.59 - 0.1 \times (1.155) = 0.4745$~~

$$\& \; C = (0.885 - 0.5)^2 = 0.148225$$

This is the 2nd row of above table (shown in next page)

Also, $w_{new} = w_{old} - 0.1 \times \frac{\partial C}{\partial w}(w_{old})$.

$$= 0.59 - 0.1 \times 1.155$$

$$= 0.4745$$

The following table can be constructed.

	$a^{(0)}$	w_{old}	b	$a^{(1)}$	C	$\frac{\partial C}{\partial w}(w_{old}) = 3(1.5w_{old} - 0.5)$	$w_{new} = w_{old} - .1\frac{\partial C}{\partial w}(w_{old})$
							0.59
Step 1	1.5	0.8	0	1.2	0.49	2.1	0.4745
Step 2	1.5	0.59	0	0.885	0.148	1.155	0.410975
Step 3	1.5	0.4745	0	0.71175	0.045	0.63525	⋮
							0.333

Repeated iteration eventually converges to 0.333 & this will be the ideal weight for which the cost will be minimum, ~~Note that In this simple case, $C = (wa^{(0)}$~~ $- y$ for this training example $(1.5, 0.5)$. Now, we note that

$$C = (1.5w - 0.5)^2 \;\&$$

$$\frac{dC}{dw} = 2(1.5w - 0.5) = 0 \;\&\; \frac{d^2C}{dw^2} = 3 > 0$$

already tells you to choose $w = \frac{1}{3}$, as C is minimum (global) in this case. However, in general, for vector valued inputs & outputs, the computations involved with equating partial derivatives to zero & evaluation Hessian are too expensive when compared to (stochastic) gradient descent algorithm.