Module-1 VECTOR CALCULUS

⇒ Function of several variables

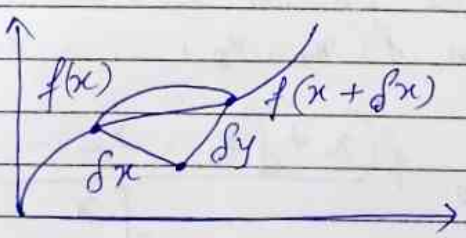
A real valued function of real variables 'n' is a function that takes as input n real numbers, commonly represented by the variables $\{x_1, x_2, x_3 \dots x_n\}$ for producing another real number. The value of the function commonly denoted as $f(x_1, x_2, x_3, \dots x_n)$

⇒ Differentiation & partial differentiation

* Differentiation of a uni-variate function

The difference quotient of a uni-variate function $y = f(x)$, $x, y \in R$ is

$$\frac{\delta y}{\delta x} = \frac{f(x+\delta x) - f(x)}{\delta x}$$



It computes the slope of secant line through two points $f(x_0)$ and $f(x_0+\delta x)$

It can also be considered the average slope of the function between x and $x+\delta x$ if we assume the function f to be a linear function for in the limit $\delta x \to 0$ we obtain the tangent of the function $f(x)$ at the point x if x is differentiable.

→ For $h>0$ the derivative of the function $f(x)$ at the point x for $h>0$ is defined as

$$\frac{df}{dx} = \lim_{h\to 0} \frac{f(x+h)-f(x)}{\delta x(h)}$$

* Secant line becomes tangent line when $h>0$

⇒ Partial differentials

For $x \in R^n$ of n variables $(x_1, x_2 \cdots x_n)$ we define the partial derivative of the function $f(x_1, x_2, \cdots x_n)$

$$\frac{\delta f}{\delta x_1} = \lim_{\delta x \to 0} \frac{f(x_1+\delta x_1, x_2 \cdots x_n)-}{\delta x}$$

$$\frac{\delta f}{\delta x_2} = \lim_{\delta x \to 0} \frac{f(x_2+\delta x_2, x_3, \cdots x_n) - f(x}{\delta x}$$

⋮

$$\frac{\partial f}{\partial x_n} = \lim_{\delta x \to 0} \frac{f(x_1, x_2 \cdots x_n + \delta x) - f(x_1, x_2 \cdots x_n)}{\delta x}$$

Similarly,

1. $\frac{\partial^2 f}{\partial x_1^2} = \frac{\partial}{\partial x_1}\left(\frac{\partial f}{\partial x_1}\right)$

2. $\frac{\partial^2 f}{\partial x_2^2} = \frac{\partial}{\partial x_2}\left(\frac{\partial f}{\partial x_2}\right)$

3. $\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_1}\left(\frac{\partial f}{\partial x_2}\right)$

4. $\frac{\partial^2 f}{\partial x_2 \partial x_1} = \frac{\partial}{\partial x_2}\left(\frac{\partial f}{\partial x_1}\right)$

$f(x_1, x_2 \cdots x_n)$

$, x_2, \cdots x_n)$

Problems 6

1. Obtain the partial derivative for $f(x,y) = (x + 2y^3)^2$

$$f(x,y) = (x + 2y^3)^2$$
$$= x^2 + (2y^3)^2 + 2(x)(2y^3)$$
$$= x^2 + 4y^6 + 4xy^3$$

$$\frac{\partial f}{\partial x} = \frac{\partial}{\partial x}(x^2 + 4y^6 + 4xy^3)$$

$$\boxed{= 2x + 4y^3}$$

$$\frac{\partial f}{\partial y} = \frac{\partial}{\partial y}(x^2 + 4y^6 + 4xy^3)$$

$$\boxed{= 24y^5 + 12xy^2}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}(2x + 4y^3)$$

$$\boxed{= 2}$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}(24y^5 + 12xy^2)$$

$= (120y^3 + 24xy)$

$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial x}(24y^3 + 12xy^2)$

$= 12y^2$

$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial y}(2x + 4y^3)$

$= 12y^2$

2. If $z = x^2y + xy^2$, $x = at$; $y = 2at$. Find $\frac{dz}{dt}$.

$z = x^2y + xy^2$

$x = at$, $y = 2at$

$z = (at)^2(2at) + (at)(2at)^2$

$= a^2t^2(2at) + (at)(4a^2t^2)$

$= 2a^3t^3 + 4a^3t^3$

$\frac{dz}{dt} = \frac{d}{dt}(2a^3t^3 + 4a^3t^3)$

$\frac{dz}{dt} = [2a^3(3t^2) + 4a^3(3t^2)]$

$$= \frac{dz}{dt} = 6a^3t^2 + 12a^3t^2$$

$$= \frac{dz}{dt} = 18a^3t^2$$

$z \to (x, y) \to t$

$$\frac{dz}{dt} = \frac{\partial z}{\partial x}\left(\frac{dx}{dt}\right) + \frac{\partial z}{\partial y}\left(\frac{dy}{dt}\right)$$

$$\frac{\partial z}{\partial x} = \frac{\partial}{\partial x}(x^2y + xy^2)$$

$$\boxed{= 2xy + y^2}$$

$$\frac{\partial z}{\partial y} = \frac{\partial}{\partial y}(x^2y + xy^2)$$

$$\boxed{= x^2 + 2xy}$$

$$\frac{dx}{dt} = \frac{d}{dt}(at) = a$$

$$\frac{dy}{dt} = \frac{d}{dt}(2at) = 2a$$

$$\frac{dz}{dt} = [2xy + y^2](a) + [x^2 + 2xy](2a)$$

$x = at \; ; \; y = 2at$

$$\frac{dz}{dt} = [2(at)(2at) + (2at)^2](a) + [a^2t^2 + 2(at)(2at)](2a)$$

$$= 4a^3t^2 + 4a^3t^2 + 2a^3t^2 + 8a^3t^2$$

$$\boxed{\frac{dz}{dt} = 18a^3t^2}$$

4. $z = x^2 + y^2$ → where $x = e^u \sin v$, $y = e^u \cos v$, find $\frac{\delta z}{\delta u}$, $\frac{\delta z}{\delta v}$

$z \Rightarrow (x, y) \rightarrow (u, v)$

$$\frac{\delta z}{\delta u} = \frac{\delta z}{\delta x}\left(\frac{dx}{du}\right) + \frac{\delta z}{\delta y}\left(\frac{dy}{du}\right)$$

$$\frac{\delta z}{\delta x} = \frac{\delta}{\delta x}(x^2 + y^2) = 2x$$

$$\frac{\delta z}{\delta y} = \frac{\delta}{\delta y}(x^2 + y^2) = 2y$$

$$\frac{dx}{du} = \frac{\delta}{\delta u}(e^u \sin v) = e^u \sin v$$

$$\frac{dy}{dv} = \frac{\delta}{\delta v}(e^u \cos v) = e^u \sin v$$

$$\frac{\delta z}{\delta u} = 2x(e^u \sin v) + 2y(e^u \sin v)$$

at $x = e^u \sin v$, $y = e^u \cos v$

$$\frac{\delta z}{\delta u} = 2(e^u \sin v)(e^u \sin v) + 2(e^u \cos v)(e^u \cos v)$$

$$= 2e^{2u} \sin^2 v + 2e^{2u} \cos^2 v$$

$$= 2e^{2u}(\sin^2 v + \cos^2 v)$$

$$= 2e^{2u}$$

$$\frac{\delta z}{\delta v} = \frac{\delta z}{\delta x}\left(\frac{\delta x}{\delta v}\right) + \frac{\delta z}{\delta y}\left(\frac{\delta y}{\delta v}\right)$$

$$\frac{\delta x}{\delta v} = \frac{\delta}{\delta v}(e^u \sin v) = e^u \cos v$$

$$\frac{\delta y}{\delta v} = \frac{\delta}{\delta v}(e^u \cos v) = -e^u \sin v$$

$$\frac{\delta z}{\delta v} = 2x(e^u \cos v) + 2y(-e^u \sin v)$$

$$= 2(e^u \sin v)(e^u \cos v) + 2(e^u \cos v)(-e^u \sin v)$$

$$= 2e^{2u} \sin v \cos v - 2e^{2u} \sin v \cos v$$

$$= 0$$

5 For $f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3 \in R$ compute the partial derivatives. Also obtain the gradient and mention its size

$$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1}\left(x_1^2 x_2 + x_1 x_2^3\right)$$

$$= 2x_1 x_2 + x_2^3$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2}\left(x_1^2 x_2 + x_1 x_2^3\right)$$

$$= x_1^2 + 3x_1 x_2^2$$

$$\frac{\partial^2 f}{\partial x_1^2} = \frac{\partial}{\partial x_1^2}\left(2x_1 x_2 + x_2^3\right)$$

$$= 2x_2$$

$$\frac{\partial^2 f}{\partial x_2^2} = \frac{\partial}{\partial x_2^2}\left(x_1^2 + 3x_1 x_2^2\right)$$

$$= 6x_1 x_2$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_1}\left(x_1^2 + 3x_1 x_2^2\right)$$

$$= 2x_1 + 3x_2^2$$

$$\frac{\partial^2 f}{\partial x_2 \partial x_1} = \frac{\partial}{\partial x_2}\left(2x_1 x_2 + x_2^3\right)$$

$$= 2x_1 + 3x_2^2$$

Gradient

$$\nabla = \frac{\partial}{\partial x}(i) + \frac{\partial}{\partial y}(j) + \frac{\partial}{\partial z}(k)$$

$$\nabla f = \frac{\partial f}{\partial x}(i) + \frac{\partial f}{\partial y}(j) + \frac{\partial f}{\partial z}(k)$$

$$\nabla f = \frac{\partial f}{\partial x_1}(i) + \frac{\partial f}{\partial x_2}(j)$$

$$= \frac{\partial}{\partial x_1}(x_1^2 x_2 + x_1 x_2^3)(i) + \frac{\partial}{\partial x_2}(x_1^2 x_2 + x_1 x_2^3)j$$

$$\nabla f = (2x_1 x_2 + x_2^3)i + (x_1^2 + 3x_1 x_2^2)j$$

writing in matrix form;

$$\boxed{2x_1 x_2 + x_2^3 \quad , \quad x_1^2 + 3x_1 x_2^2}$$

Therefore, the size of gradient is (1×2).

6. Let $f(x_1, x_2) = x_1^2 + 2x_2$ where $x_1 = \sin t$, $x_2 = \cos t$. Find $\frac{df}{dt}$. Also find the gradient.

$f \to (x_1, x_2) \to (t)$

$$\frac{df}{dt} = \frac{\partial f}{\partial x_1}\left(\frac{dx_1}{dt}\right) + \frac{\partial f}{\partial x_2}\left(\frac{dx_2}{dt}\right)$$

$$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1}(x_1^2 + 2x_2) = 2x_1$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2}(x_1^2 + 2x_2) = 2$$

$$\frac{dx_1}{dt} = \frac{d}{dt}(\sin t) = \cos t$$

$$\frac{dx_2}{dt} = \frac{d}{dt}(\cos t) = -\sin t$$

$$\frac{df}{dt} = 2x_1(\cos t) + 2(-\sin t)$$

$$\frac{df}{dt} = 2(\sin t)(\cos t) + 2(-\sin t)$$

$$\boxed{\frac{df}{dt} = 2\sin t \cos t - 2\sin t}$$

Gradient

$$\nabla f = \left(\frac{\partial f}{\partial x_1}\right) i + \left(\frac{\partial f}{\partial x_2}\right) j$$

$$\nabla f = \frac{\partial}{\partial x_1}\left(x_1^2 + 2x_2\right) i + \frac{\partial}{\partial x_2}\left(x_1^2 + 2x_2\right) j$$

$$\nabla f = (2x_1)\, i + 2j$$

Matrix form $\Rightarrow$ $[2x_1 \;,\; 2]$

The size of gradient is (1×2).

7. $u = xy + yz + zx$ at $x = t\cos t$, $y = t\sin t$ and $z = t$. Find $\frac{du}{dt}$

$u \rightarrow (x, y, z) \longrightarrow (t)$

$$\frac{du}{dt} = \frac{\partial u}{\partial x}\cdot\frac{dx}{dt} + \frac{\partial u}{\partial y}\cdot\frac{\partial y}{\partial t} + \frac{\partial u}{\partial z}\cdot\frac{\partial z}{\partial t}$$

$$\frac{\partial u}{\partial x} = \frac{\partial}{\partial x}\left(xy + yz + zx\right)$$

$$= y + z$$

$$\frac{\partial u}{\partial y} = \frac{\partial}{\partial y}\left(xy + yz + zx\right)$$

$$= x + z$$

$$\frac{\partial u}{\partial z} = \frac{\partial}{\partial z}(xy + yz + zx)$$

$$= y + x$$

$$\frac{\partial x}{\partial t} = \frac{d}{dt}(t\cos t) = -\sin t$$

$$\frac{dy}{dt} = \frac{d}{dt}(t\sin t) = \cos t$$

$$\frac{dz}{dt} = \frac{d}{dt}(t) = 1$$

$$\frac{du}{dt} = (y+z)(-\sin t) + (x+z)(\cos t) + (y+x)(1)$$

$$= (t\sin t + t)(-\sin t) + (t\cos t + t)(\cos t) + (t\cos t + t\sin t)$$

$$= -t\sin^2 t - t\sin t + t\cos^2 t + t\cos t + t\cos t + t\sin t$$

$$= t\cos^2 t - t\sin^2 t + 2t\cos t$$

$$= t(\cos^2 t - \sin^2 t) + 2t\cos t$$

→ Gradient of Vector valued functions

For a function $f : R^n \to R^m$ and a vector $x = (x_1, x_2, \dots x_n)^T \in R^n$ the corresponding vector of function values is given by $f(x) = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}^T \in R^m$

The partial derivatives of a vector valued function with respect to $x_i \in R$ where $i = 1$ to n is given as the vector $\frac{\partial f}{\partial x_i} =$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_i} \\ \\ \frac{\partial f_2}{\partial x_2} \\ \vdots \\ \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$$\lim_{h \to 0} \frac{f(x_1, x_2 \dots x_i + h) - f(x_i)}{h}$$

$$\lim_{h \to 0} \frac{f(x_i, x_2, \dots x_i + h) - f(x_i)}{h}$$

$$\lim_{h \to 0} \frac{f(x_i, x_2, \dots x_i + h) - f(x_i)}{h}$$

we know that the gradient of f with respect to a vector is the row vector of the partial derivative. Every partial derivative $\frac{\partial f_i}{\partial x_i}$ is itself a column vector. we obtain the gradient of $f: \mathbb{R}^n \to \mathbb{R}^m$ w.r.t $x \in \mathbb{R}^n$ by collecting these partial derivatives

$$\frac{\partial f}{\partial x}(x) = \left[\frac{\partial f}{\partial x_1}(x) + \frac{\partial f}{\partial x_2}(x) + \cdots \frac{\partial f}{\partial x_n}(x)\right]$$

$$= \frac{\partial f_1(x_1)}{\partial x_1} \quad \frac{\partial f_2(x_2)}{\partial x_2} \quad \cdots \quad \frac{\partial f_2(x_n)}{\partial x_n}$$

$$\begin{matrix} \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_n} \end{matrix} \in \mathbb{R}^{m \times n}$$

The gradient of a function $f: \mathbb{R}^n \to \mathbb{R}^m$ is a matrix of size $m \times n$ is called the 'Jacobians'.

i.e $J = \Delta_x f = \frac{\partial f(x)}{\partial x}$

→ The gradient of matrix with respect to a vector

Let $A \in R^{4\times 2}$ ← matrix of order 4×2
and $x \in R^3$ ← vector in 3 dimensional space i.e $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

Partial derivatives $\frac{\partial A}{\partial x_1} \in R^{4\times 2}$, $\frac{\partial A}{\partial x_2} \in R^{4\times 2}$

and $\frac{\partial A}{\partial x_3} \in R^{4\times 2}$

→ ∴ Gradient of a matrix w.r.t a vector $\frac{\partial A}{\partial x} \in R^{4\times 2\times 3}$

Let $f = Ax$; $f \in R^n$; $A \in R^{m\times n}$ and $x \in R^n$

Gradient of f is the collection of all partial derivatives

i.e $\frac{\partial f}{\partial A} = \begin{bmatrix} \frac{\partial f_1}{\partial A} \\ \frac{\partial f_2}{\partial A} \\ \vdots \\ \frac{\partial f_n}{\partial A} \end{bmatrix} \in R^{m\times(m\times n)}$

where $\frac{\partial f_i}{\partial A} \in R^{1 \times (m \times n)}$

In general, $f_i = \sum_{j=1}^{N} A_{ij} x_j$, $i = 1, 2, \ldots m$.

Then, the partial derivative of f w.r.t a row of A

$\frac{\partial f_i}{\partial A_q} = x_q \Rightarrow \frac{\partial f_i}{\partial A_i} = x^T \in R^{1 \times 1 \times N}$ and

$\frac{\partial f_i}{\partial A_{k \neq i}} = 0^T \in R^{1 \times 1 \times N}$

Since f_i maps onto R and each row of A is of size $1 \times N$ we obtain a $(1 \times 1 \times 6)$ sized terms as the partial derivatives of f_i w.r.t a row of A

$$\frac{\partial f_i}{\partial A} = \begin{bmatrix} 0^T \\ \vdots \\ 0^T \\ x^T \\ 0^T \\ \vdots \\ 0^T \end{bmatrix} \in R^{1 \times (m \times N)}$$

⇒ Gradient of matrices w.r.t matrices

Consider a matrix $A \in R^{M \times N}$ and $f = R^{M \times N} \rightarrow R^{N \times N}$ with $f(A) = A^T A = K \in R^{M \times N}$

The Gradient $\frac{\partial K}{\partial A}$ is given by,

$$\frac{\partial K}{\partial A} \in A^{(N \times N) \times (M \times N)}$$

$$\Rightarrow \frac{\partial K_{pq}}{\partial A} \in R^{1 \times M \times N}$$

For $p, q = 1, 2, \ldots N$ where K_{pq} is the $(p,q)^{th}$ entry of $K = f(A)$

$$K_{pq} = r_p T_{rq} = \sum_{m=1}^{M} A_{mp} \cdot A_{mq}$$

then

$$\frac{\partial K_{pq}}{\partial A_{ij}} = \sum_{m=1}^{M} \frac{\partial}{\partial A_{ij}} (A_{mp} \cdot A_{mq}) = \gamma_{pqij}$$

where $\gamma_{pqij} = A_{iq}$ if $j=p$ and $p \neq q$

A_{ip} if $j=q$ and $p \neq q$

$2A$ if $j=p$ and $p=q$

0, otherwise.

Problems on Jacobians :-

1. If $x, y \in R^2$ and $y_1 = -2x_1 + x_2$ $y_2 = x_1 + x_2$. Show that the Jacobian determinant $|\det(J)| = 3$

Sol^n:- $y_1 = -2x_1 + x_2$ $\quad y_2 = x_1 + x_2$

$$J = \begin{vmatrix} \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{vmatrix}$$

$$J = \begin{vmatrix} \frac{\partial}{\partial x_1}(-2x_1 + x_2) & \frac{\partial}{\partial x_2}(-2x_1 + x_2) \\ \frac{\partial}{\partial x_1}(x_1 + x_2) & \frac{\partial}{\partial x_2}(x_1 + x_2) \end{vmatrix}$$

$$J = \begin{vmatrix} -2 + 0 & 0 + 1 \\ 1 + 0 & 0 + 1 \end{vmatrix}$$

$$J = \begin{vmatrix} -2 & 1 \\ 1 & 1 \end{vmatrix}$$

$$= |-2 - 1|$$

$$= |-3|$$

$$J = 3 \quad //$$

2. Find the Jacobian determinant of

$y_1 = x_1^2 + x_2^2 + x_3^2 \quad y_2 = x_1x_2 + x_2x_3 + x_3x_1$

$y_3 = x_1 + x_2 + x_3$

Soln: $J = \begin{vmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} \end{vmatrix}$

$J = \begin{vmatrix} 2x_1 & 2x_2 & 2x_3 \\ x_2 + x_3 & x_1 + x_3 & x_1 + x_2 \\ 1 & 1 & 1 \end{vmatrix}$

$\because$ Since the R_3 is having same elements

$\Rightarrow J = 0$ //.

* Product Rule :-

$$\frac{d}{dx}(f(x) \cdot g(x)) = f(x) \cdot \frac{d}{dx}(g(x)) + g(x) \cdot \frac{d}{dx}(f(x))$$

* Sum Rule :-

$$\frac{d}{dx}(f(x) + g(x)) = \frac{d}{dx}(f(x)) + \frac{d}{dx}(g(x))$$

* Chain Rule :-

$$\frac{d}{dx}(fog)(x) = \frac{d}{dx}(g\,f(x)) = \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial x}$$

3. If a function $h: R \to R$, $R(t) = fog(t)$ with $f: R^2 \to R$, $g: R \to R^2$ and $f(x) = -e^{x_1 x_2^2}$, $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} =$

$g(t) = t\cos t \; t\sin t$. Compute gradient of h with respect to t.

Soln $\frac{\partial f}{\partial x} \in R^{1\times 2}$, $\frac{\partial g}{\partial x} \in R^{2\times 1}$

$f(x) = e^{x_1 x_2^2}$

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} t\cos t \\ t\sin t \end{bmatrix}$

$x_1 = t\cos t$

$x_2 = t\sin t$

$\frac{\partial h}{\partial t} = \frac{\partial f}{\partial x} \cdot \frac{\partial x}{\partial t}$

$= \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{bmatrix}$

$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1}\left(e^{x_1 x_2^2}\right)$

$= e^{x_1 x_2^2} \cdot x_2^2$

$$\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2}\left(e^{x_1 x_2^2}\right) = 2e^{x_1 x_2} \cdot e^{x_1 x_2^2}$$

$$\frac{\partial x_1}{\partial t} = \frac{\partial}{\partial t}(t \cos t)$$

— product rule —

$$\frac{\partial x_1}{\partial t} = \cos(t) - (t)\sin t$$

$$\frac{\partial x_2}{\partial t} = \frac{\partial}{\partial t}((t)\sin t)$$

$$= \sin t + (t)\cos t$$

$$\frac{\partial h}{\partial t} = \begin{bmatrix} e^{x_1 x_2^2} \cdot x_2^2 & 2e^{x_1 x_2} \cdot e^{x_1 x_2^2} \end{bmatrix} \cdot \begin{bmatrix} \cos t - t \sin t \\ \sin t + t \cos t \end{bmatrix}$$

If $f(x) = Ax$, where $f(x) \in \mathbb{R}^M$, $A \in \mathbb{R}^{M \times N}$, $x \in \mathbb{R}^N$, obtain gradient and mention its size.

Soln $f : \mathbb{R}^N \longrightarrow \mathbb{R}^M$

$y = f(x)$

Range $\mathbb{R}^N$ — Domain $\in \mathbb{R}^M$.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} = \begin{bmatrix} f_1(x_1 \cdots x_n) \\ f_2(x_1 \cdots x_n) \\ \vdots \\ f_n\, x_1 \cdots x_n \end{bmatrix}$$

gradient collection of partial derivative

$$\begin{bmatrix} \frac{dy_1}{dx} \\ \frac{dy_2}{dx} \\ \vdots \\ \frac{dy_n}{dx} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$f(x) = Ax \quad ; \quad A \in R^{M \times N} \quad x \in R^N$
$f(x) = y \in R^M$

Obtain the gradient of scalar
$\phi = 4x_0 + 2x_1 - 3x_2 + x_3$
with respect to the matrix

$$X = \begin{bmatrix} x_0 & x_1 \\ x_2 & x_3 \end{bmatrix}$$

$$x^T = \begin{bmatrix} x_0 & x_2 \\ x_1 & x_3 \end{bmatrix}$$

$$\nabla\phi = \frac{\partial \phi}{\partial x}$$

$$\frac{\partial}{\partial x} = \begin{bmatrix} \frac{\partial \phi}{\partial x_0} & \frac{\partial \phi}{\partial x_2} \\ \frac{\partial \phi}{\partial x_1} & \frac{\partial \phi}{\partial x_3} \end{bmatrix}$$

$$\frac{\partial \phi}{\partial x} = \begin{bmatrix} 4 & -3 \\ 2 & 1 \end{bmatrix} = 10$$

Size of the matrix is 2×2

Obtain the gradient of scalar $y = x_0 + x_1 + 4x_2 - 6x_3$ w.r.t matrix $\begin{bmatrix} x_0 & x_1 \\ x_2 & x_3 \end{bmatrix}$

$$x^T = \begin{bmatrix} x_0 & x_2 \\ x_1 & x_3 \end{bmatrix}$$

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y}{\partial x_0} & \frac{\partial y}{\partial x_2} \\ \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_3} \end{bmatrix}$$

$$\frac{\partial y}{\partial x} = \begin{bmatrix} 1 & 4 \\ 1 & -6 \end{bmatrix} = 10.$$

Size of the matrix is 2×2

obtain the gradient of vector

$$f = \begin{bmatrix} e^{x_0 x_1} & e^{x_2 x_3} \end{bmatrix}$$ w.r.t

the matrix $\begin{bmatrix} x_0 & x_1 \\ x_2 & x_3 \end{bmatrix}$

$$x^T = \begin{bmatrix} x_0 & x_2 \\ x_1 & x_3 \end{bmatrix}$$

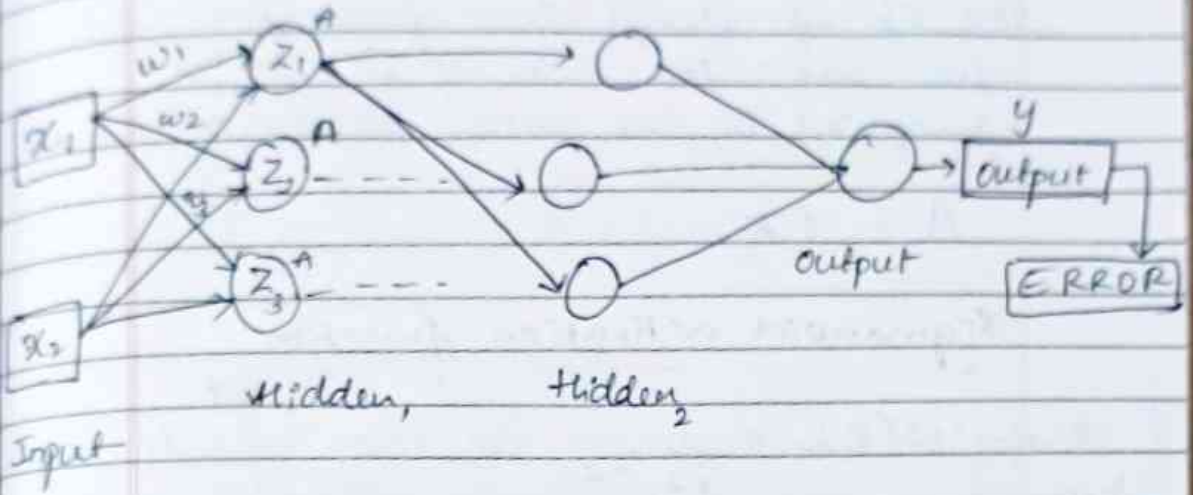
$$\frac{\partial \vec{f}}{\partial x} = \begin{bmatrix} \frac{\partial \vec{f}}{\partial x_0} & \frac{\partial \vec{f}}{\partial x_2} \\ \frac{\partial \vec{f}}{\partial x_1} & \frac{\partial \vec{f}}{\partial x_3} \end{bmatrix}$$

$$\frac{\partial \vec{f}}{\partial x} = \begin{bmatrix} \begin{bmatrix} e^{x_0 x_1} \cdot x_1 & 0 \end{bmatrix} & \begin{bmatrix} 0 & e^{x_2 x_3} \cdot x_3 \end{bmatrix} \\ \begin{bmatrix} e^{x_0 x_1} \cdot x_0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & e^{x_2 x_3} \cdot x_2 \end{bmatrix} \end{bmatrix}$$

The above matrix is the gradient vector.

Module - 2

Back Propagation



The whole learning process is divided into a recursive, forward & back-ward propagation.

Forward propagation

Inputs are passed to neurons of hidden layer with some randomly initialized weights along biases (w_1 w_2 w_3) as a linear transformation shown below.

$Z = (input * weight) + bias$

to solve complex problems, non-linear

transformation is introduced to be achieved by an activation function.

The o/p of linear transformation (z) i.e weighted sum of inputs is supplied to the below activation function

$$A = f(z)$$

Sigmanoid activation function

$$S(z) = \frac{1}{1+e^{-z}}$$

Every layer ~~funct~~ applies this linear followed by non-linear equations hence, final o/p of every layer would be the o/p of activation function.

We get predicted o/p in an o/p layer (y) doing same processes in every layer

Backward propagation.

Predicted o/p (y) may differ from the actual o/p hence loss is calculated using loss function i.e 'J'

This tells that how diverted our prediction is from the actual.

Let's say loss using mean sum of squared loss function.

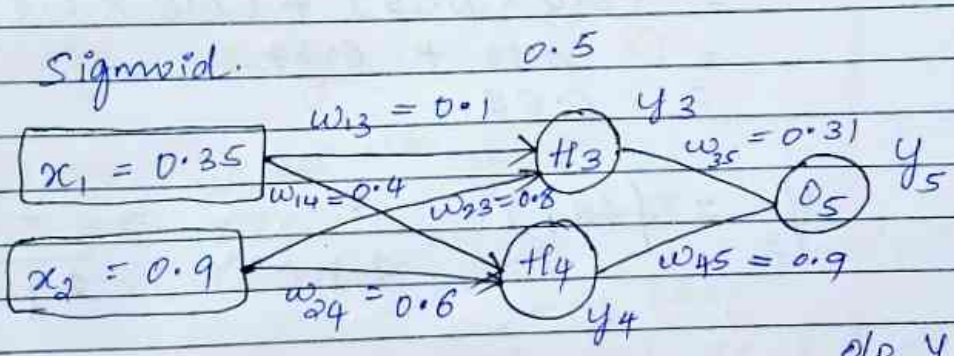
i.e defined as

$$J = \text{loss} = \sum_{1}^{N} (y - \hat{y})$$

(y → actual o/p, $\hat{y}$ → predicted o/p)

Problems:-

1. Assume that the neurons have sigmoid activation function. Perform a forward pass and a backward pass on the network. Assume that the actual o/p of Y is 0.5 and learning rate is 1. Perform another forward pass. and write the given network also.



Sol"n: Forward pass (To calculate o/p y_3, y_4, y_5)

To calculate y_3, y_4

$$= a_j = \sum_j (w_{ij} \times x_i)$$

Sigmoid activation function $= y_j = F(a_j) = \frac{1}{1+e^{-a_j}}$

$y_3 = a_1 = (w_{1,3} \times x_1) + (w_{23} \times x_2)$

$= (0.1 \times 0.35) + (0.8 \times 0.9)$

$= 0.035 + 0.72$

$= 0.755$

$y_4 = a_2 = (w_{14} \times x_1) + (w_{24} \times x_2)$

$= (0.4 \times 0.35) + (0.6 \times 0.9)$

$= 0.14 + 0.54$

$= 0.68$

$y_3 = f(a_1) = \frac{1}{1+e^{-a_1}} = \frac{1}{1+e^{-0.755}}$

$= 0.68$

$y_4 = f(a_2) = \frac{1}{1+e^{-0.68}} = 0.6637$

$y_s = a_s = (0.38 \times 0.68) + (0.7 \times 0.6611)$

$= 0.2584 + 0.5743$

$= 0.8013$

$y_s = f(a_s) = \frac{1}{1 + e^{-0.8013}} = 0.6903$

$\Rightarrow$ output $= y_s = 0.6903 \approx 0.69$

$\Rightarrow$ Error $= (\underset{y_{target}}{\text{Target o/p}} - \underset{y_s}{\text{o/p obtained}})$

$= 0.5 - 0.69$

$= -0.19$

- Updating weights to reduce the error -

$\Delta w_{ji} = \eta \delta_j o_i$

$o_i \rightarrow$ o/p at i^{th} unit

$\delta_j \rightarrow$ error at j^{th} unit

$\eta \rightarrow$ learning rate

$\delta_j = o_j (1 - o_j)(t_j - o_j)$ If J is an output unit (t_j → target output)

$\delta_j = o_j (1 - o_j) \sum_k \delta_k w_{kj}$ If J is a hidden unit

→ k is the no. of outputs.

Backward pass

Compute δ_3 δ_4 δ_5 at y_3 y_4 y_5

For Output unit 6

$$\delta_5 = O_5(1-O_5)(t_5-O_5)$$

$$= 0.6903\,(1-0.6903)(0.5-0.6903)$$

$$= 0.6903\,(0.3097)(-0.19)$$

$$= -0.0406$$

For hidden unit 8-

$$\delta_3 = H_3(1-H_3) * w_{35} * \delta_5$$

$$= 0.68(1-0.68) * 0.3 * -0.0406$$

$$= -0.0027$$

$$\delta_4 = H_4(1-H_4) * w_{45} * \delta_5$$

$$= 0.6637\,(1-0.6637) * 0.9 * -0.0406$$

$$= -0.0082$$

$\Delta w_{45} = \eta \delta_5 y_4$

$= 1 \times -0.0406 \times 0.6637$

$= -0.0269$

(updated) $w_{45} = \Delta w_{45} + w_{45}(old)$

$= -0.0269 + 0.9$

$= 0.8731$

$\Delta w_{14} = \eta \delta_4 x_1 = 1 \times -0.0082 \times 0.35$

$= -0.0029$

(updated) $w_{14} = -0.0029 + 0.4$

$= 0.3971$

$\Delta w_{13} = \eta \delta_3 x_1 = 1 \times -0.0027 \times 0.35$

$= -0.0009$

(updated) $w_{13} = 0.0991$

(updated) $w_{23} = 0.7976$

" $w_{24} = 0.5926$

" $w_{35} = 0.2724$

Forward pass

$a_1 = (w_{13} \times x_1) + (w_{23} \times x_2)$

$= (0.0991 \times 0.35) + (0.7976 \times 0.9)$

$= 0.7525$

$y_3 = 1/1+e^{-0.7525} = 0.6797$

$$a_2 = (w_{14} \times x_1) + (w_{24} \times x_0)$$

$$= 0.6723$$

$$y_4 = f(a_2) = \frac{1}{1 + e^{-0.6723}} = 0.6620$$

$$a_3 = (w_{35} \times y_3) + (w_{45} \times y_4)$$

$$= 0.7631$$

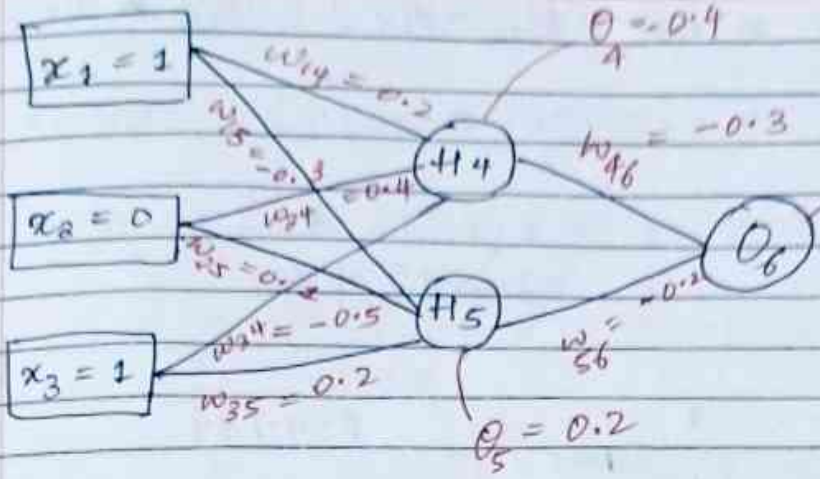
$$y_5 = f(a_3) = \frac{1}{1 + e^{-0.7631}} = 0.6820$$

$$\text{Error} = y_{target} - y_5$$

$$= 0.5 - 0.6820$$

$$= -0.1820.$$

2. Assume that the neurons have a sigmoid activation function. Perform a forward pass & backward pass on the network. Assume that the actual o/p of $y = 1$ and learning rate is 0.9. perform another forward pass.



Soln: Forward Pass:-

$$H_4 = a_1 = (w_{14} \times x_1) + (w_{24} \times x_2) + (w_{34} \times x_3)$$

$$= (0.2 \times 1) + (0.4 \times 0) + (-0.5 \times 1)$$

$$= 0.2 + (-0.5)$$

$$= -0.3 - 0.4$$

$$H_4 = f(a_1) = \frac{1}{1+e^{-a_1}} = \frac{1}{1+e^{+0.7}} = 0.3318$$

$$H_5 = a_2 = (w_{15} \times x_1) + (w_{25} \times x_2) + (w_{35} \times x_3)$$

$$= (-0.3 \times 1) + (0.1 \times 0) + (0.2 \times 1)$$

$$= -0.3 + 0.2$$

$$= -0.1 + 0.2$$

$$H_5 = f(a_2) = \frac{1}{1+e^{-+a_2}} = \frac{1}{1+e^{-0.1}} = 0.5250$$

Evaluating biases :- $\Rightarrow H_4 = 0.3318$

$\Rightarrow H_5 = 0.5250$

$O_6 = a_3 = (w_{46} \times h_4) + (w_{56} \times h_5)$

$= (-0.3 \times 0.2318) + (-0.2 \times 0.5250)$

$= -0.0995 + (-0.1050)$

$= -0.2045 + 0.1$

$O_6 = \frac{1}{1+e^{0.1045}} = 0.4739$

$\Rightarrow \boxed{O_6 = 0.4739}$

Error = Target o/p − obtained o/p

$= 1 - 0.4739$

$\Rightarrow \boxed{\text{Error} = 0.5261}$

Backward pass

– Computing δ_4 δ_5 δ_6 —

For output unit 6

$\delta_6 = O_6(1-O_6)(t_6 - O_6)$

$= 0.4739(1-0.4739)(1-0.4739)$

$= 0.4739(0.5261)(0.5261)$

$= 0.1312$

For hidden unit 6

$$\delta_4 = H_4 (1 - H_4) * w_{46} * \delta_6$$

$$= 0.3318 (1 - 0.3318) * (-0.3) * 0.1312$$

$$= 0.3318 (0.6682) * (-0.3) * 0.1312$$

$$= -0.0087$$

$$\delta_5 = H_5 (1 - H_5) * w_{56} * \delta_6$$

$$= 0.5250 (1 - 0.5250) * (-0.2) * 0.1312$$

$$= 0.5250 (0.4750) * (-0.2) * 0.1312$$

$$= -0.0065$$

$$\Delta w_{46} = \eta \delta_6 H_4$$

$$= 0.9 \times 0.1312 \times 0.3318$$

$$= 0.0392$$

updated $w_{46} = 0.0392 + (-0.3)$

$$= -0.2608$$

$$\Delta w_{14} = \eta \delta_4 x_1 = 0.9 \times -0.0087 \times 1$$

$$= -0.0078$$

updated $w_{14} = -0.0078 + (0.2)$

$$= 0.1922$$

$$\Delta w_{25} = \eta \delta_5 x_2 = 0.9 \times -0.0065 \times 0$$

$$= 0$$

updated $w_{25} = 0.1$

Similarly:

$w_{14} = 0.192$ $\quad \Delta w_{14} = -0.0078$

$w_{15} = -0.306$ $\quad \Delta w_{15} = -0.0059$

$w_{24} = 0.4$ $\quad \Delta w_{24} = 0$

$w_{25} = 0.1$ $\quad \Delta w_{25} = 0$

$w_{34} = -0.508$ $\quad \Delta w_{34} = -0.0078$

$w_{35} = 0.194$ $\quad \Delta w_{35} = -0.0059$

$w_{46} = -0.261$ $\quad \Delta w_{46} = 0.0392$

$w_{56} = -0.138$ $\quad \Delta w_{56} = -0.0619$

— Changing bias value :—

$$\Delta\theta_6 = \eta \delta_6 = 0.9 \times 0.1312$$
$$= 0.1181$$

$$\theta_6 = \Delta\theta_6 + (\text{old})\,\theta_6$$
$$= 0.1181 + 0.1$$
$$= 0.2181$$

$$\Delta\theta_4 = \eta \delta_4 = 0.9 \times -0.0087$$
$$= -0.0078$$

$$\theta_4 = \Delta\theta_4 + (\text{old})\,\theta_4$$
$$= -0.0078 - 0.4$$
$$= -0.4078$$

$$\Delta\theta_5 = \eta \delta_5 = 0.9 \times -0.0065$$
$$= -0.0059$$

$\theta_5 = \Delta\theta_5 + (\text{old})\,\theta_5$

$= -0.0059 + 0.2$

$= 0.1942$

Updated bias :-

$\theta_4 = -0.40078$

$\theta_5 = 0.1942$

$\theta_6 = 0.2181$

Forward pass :-

$\Rightarrow H_4' = (w_{14} \times x_1) + (w_{24} \times x_2) + (w_{34} \times x_3)$

$= (0.192 \times 1) + (-0.508 \times 1)$

$= 0.192 - 0.508$

$= -0.3160 + (-0.4078) = \underline{-0.7238}$

$\Rightarrow H_5' = (w_{15} \times x_1) + (w_{35} \times x_3)$

$= -0.306 + 0.194$

$= -0.1120 + 0.1942 = \underline{0.0822}$

$\theta_6' \neq w_{46} \times H$

$\Rightarrow y_4' = \frac{1}{1+e^{-.3160}} = 0.4217 \Rightarrow \underline{0.3266}$ (after bias+)

$\Rightarrow y_5' = \frac{1}{1+e^{-.1120}} = 0.4720 \Rightarrow \underline{0.5205}$ (after bias+)

$\Rightarrow O_6' = (w_{46} \times y_4') + (w_{56} \times y_5')$

$= (-0.261 \times 0.4217) + (-0.138 \times 0.4720)$

$= -0.1571 + 0.2181$

$= \underline{0.0610}$

$\Rightarrow$ $\boxed{y_6' = 0.5152.}$

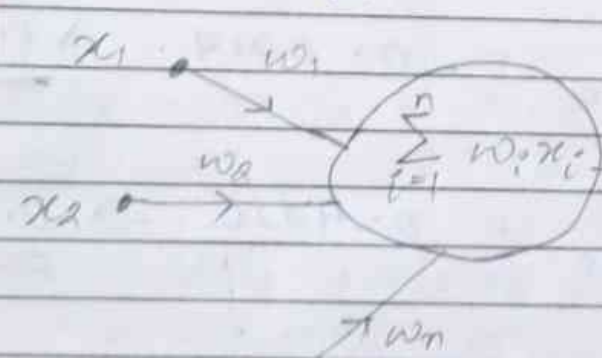
Error = $1 - 0.5152$

$\therefore$ $\boxed{\text{Error} = 0.4848.}$

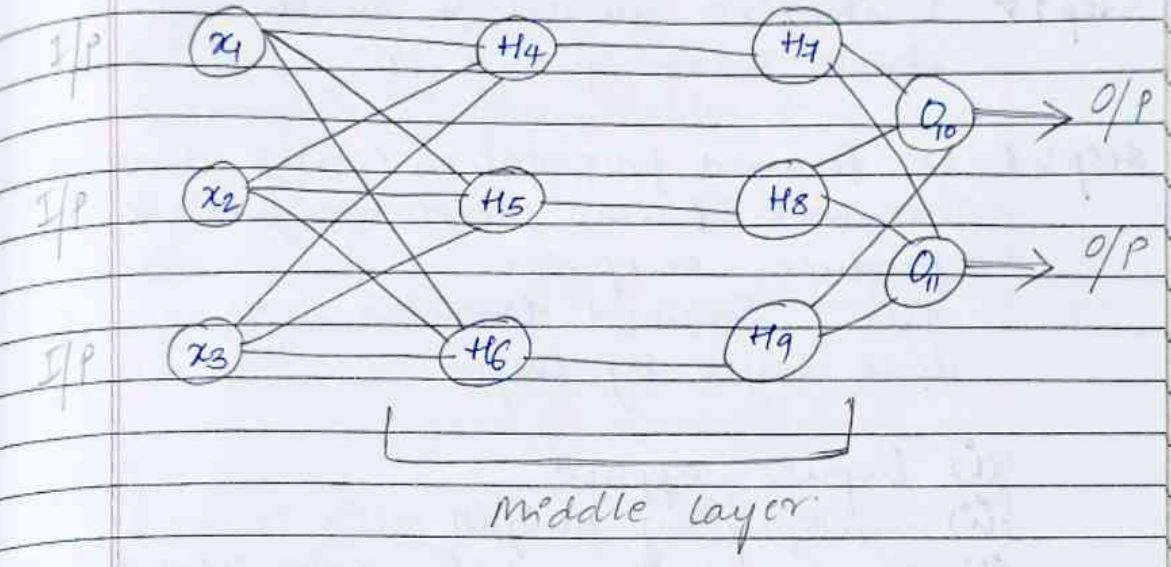
Explain Neural Networks &

* Neural networks are made up of many artificial neurons.
* Neuron can have any no. of inputs from 1 to n.
  $(x_1, x_2, \ldots x_n)$
* Each input into the neuron has its own weight $(w_1, w_2 \ldots w_n)$.
* o/p is equal to $(x_1w_1 + x_2w_2 + \ldots + x_nw_n)$

$$\text{Output} = \sum_{i=1}^{n} x_i w_i .$$



Multi-layer Network :



Sigmoid Activation function :

1. $f(a_j) = \frac{1}{1 + e^{-a_j}}$

2. tan h x : $f(x_i) = \frac{1 + e^{-a_i}}{1 + e^{a_i}}$

Back propagation Algorithm :-

Step 1:- Initialize weights @ random, choose learning rate 'n'

Step 2:- Do forward propagation (pass) through network (with fixed weights) to produce outputs.
Here, forward direction will be done layer by layer

(i) inputs applied
(ii) multiply by weights
(iii) sum and squared by activation function
(iv) Output passed to each neuron in next layer.

Repeat all the above until network o/p is produced

Step 3:- Back propagation and error calculation

(i) Compute the error (Δ or local gradient) for each output unit Δk.

(ii) Compute error for each hidden unit Δj by back propagation errors layer by layer

Step 4:- Update all the weights w_{ij} by gradient descend and go back to step 2 and repeat until the network training completion

Automatic differentiation :-

This is spl case of back propagation

Consider,

$$f = x^2(x+y) \text{ where } c = x^2$$

$$s = x+y$$

$$f = x^2(x+y) \quad , \quad F = c(s)$$

$$\frac{\partial f}{\partial x} = \frac{\partial F}{\partial c} \cdot \frac{\partial c}{\partial x} + \frac{\partial F}{\partial s} \cdot \frac{\partial s}{\partial x}$$

$$= s \cdot (2x) + c(1)$$

$$\Rightarrow \frac{\partial f}{\partial x} = 2xs + c$$

$$\frac{\partial f}{\partial y} = \frac{\partial F}{\partial c} \cdot \frac{\partial c}{\partial y} + \frac{\partial F}{\partial s} \cdot \frac{\partial s}{\partial y}$$

$$= s(0) + c(1)$$

$$\Rightarrow \frac{\partial f}{\partial y} = c$$

Gradients in a deep network :

* The function value 'y' is computed as a many level function composition.

$$y = (f_k \circ f_{k-1} \circ \cdots \circ f_1)(x)$$

w.k.t $f \circ g = f(g(x))$

$\Rightarrow$ For eg:- $f(x) = x^2$

$g(x) = x+1$

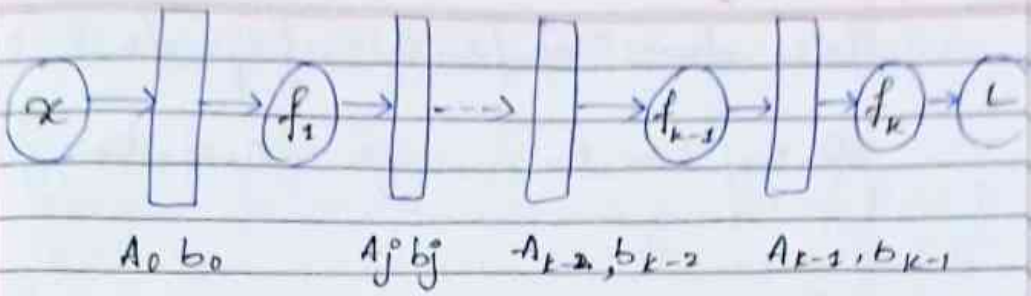
$$f \circ g = f[g(x)]$$
$$= f(x+1)$$
$$= (x+1)^2$$

Hence, similarly $y = (f_k(f_{k-1}(\cdots(f_1(x))$

* Here, x are inputs or images and y are observations.

f_i ; $i = 1, \ldots k$. posseses its own parameters.

* Forward pass is a multilayer neural network to compute the loss "L" as a function of input x and the parameters Ai , bi



Here,

$$f_i(x_{i-1}) = \sigma(A_{i-1} x_{i-1} + b_{i-1})$$

$f_i \rightarrow$ function in the i^{th} layer

$x_{i-1} \rightarrow$ Output of the layer $i-1$

$\sigma \rightarrow$ Activation function logistic sigmoid $\frac{1}{1+e^{-x}}$

$$\tanh x = \frac{1+e^{-x}}{1+e^{+x}}$$

ReLU → Rectified Linear Unit

In order to train the models, we require the gradient of Loss function 'L' w.r.t to all model parameters A_j, b_j. where $j = 1, 2, \cdots k$.

$f_0 = x \rightarrow$ input

$f_i = \sigma_i (A_{i-1} f_{i-1} + b_{i-1})$ for $i = 1, 2 \cdots k$

TO find A_j, b_j for $j = 0, 1, 2 \cdots k-1$. we have to minimize the loss function ~~$L(\theta) = \|\phi\| - f_k(\theta_1, x)$~~

Loss function $L(\theta) = \| y - f_K(\theta, x) \|^2$

$\theta = \{ A_0, b_0, \ldots\ldots A_{K-1}, b_{K-1} \}$

By definition of maxima or minima, we require the partial derivatives of L w.r.t parameters.

$\theta_j = (A_j, b_j) \; ; j = 0, 1, \ldots K-1$

By chain rule we have,

$$\frac{\partial L}{\partial \theta_{K-1}} = \frac{\partial L}{\partial f_K} \cdot \frac{\partial f_K}{\partial \theta_{K-1}}$$

$$\frac{\partial L}{\partial \theta_{K-2}} = \frac{\partial L}{\partial f_K} \cdot \underbrace{\frac{\partial f_K}{\partial \theta_{K-1}}}_{\text{w.r.t inputs}} \cdot \underbrace{\frac{\partial \theta_{K-1}}{\partial \theta_{K-2}}}_{\text{partial derivative of o/p layer w.r.t parameters}}$$

$$\frac{\partial L}{\partial \theta_{K-3}} = \frac{\partial L}{\partial f_K} \cdot \frac{\partial f_K}{\partial \theta_{K-1}} \cdot \frac{\partial \theta_{K-1}}{\partial \theta_{K-2}} \cdot \frac{\partial \theta_{K-2}}{\partial \theta_{K-3}}$$

$$\vdots$$

$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial f_K} \cdot \frac{\partial f_K}{\partial \theta_{K-1}} \cdots\cdots \frac{\partial \theta_{i+2}}{\partial \theta_{i+1}} \cdot \frac{\partial \theta_{i+1}}{\partial \theta_i}$$

Optimization using second derivative test :-

A necessary condition for x_0 to be an extreme point (max or min point) of $f(x)$ is the $\delta f(x_0) = 0$ where $x \in R^n$ and $x_0 \in R^n$ is called critical point / stationary point. A sufficient condition for a stationary point x_0 of $f(x)$ to be an extreme (maximum or minimum).

→ f has relative minimum if $f_{xx}(x_0) f_{yy}(x_0) - f^2_{xy}(x_0)$ is greater than 0. & $f_{xx}(x_0) > 0$

→ f has relative maximum if $f_{xx}(x_0) f_{yy}(x_0) - f^2_{xy}(x_0) > 0$ and $f_{xx}(x_0) < 0$.

→ f has a saddle point (neither max. nor min.) if $f_{xx}(x_0) f_{yy}(x_0) - f^2_{xy}(x_0) < 0$.

→ The test is inconclusive if $f_{xx}(x_0) f_{yy}(x_0) - f^2_{xy}(x_0)$.

Hessian Matrix form :-

A sufficient condition for an stationary point x_0 of $f(x_0)$ to be an extremum, f has :

→ f has relative minimum if $H(x_0)$ is positive definite

→ f has relative maximum if $H(x_0)$ is negative definite

→ f is a saddle point if H(x₀) is indefinite

→ The test is inconclusive if otherwise

Problems:-

1. Using the Hessian matrix classify the relative extremum for $f(x_1, x_2)$

$$\frac{1}{3}x_1^3 + x_1x_2^2 - 8x_1x_2 + 3.$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2^2 - 8x_2 \\ 2x_1x_2 - 8x_1 \end{bmatrix}$$

$$\nabla f \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix} = 0$$

$$= x_1^2 + x_2^2 - 8x_2 = 0 \qquad 2x_1x_2 - 8x_1 = 0.$$

$$2x_1x_2 = 8x_1$$

$$x_2 = \frac{8x_1}{2x_1} \quad 4$$

$$x_2 = 4$$

$$\Rightarrow x_1 = 0$$

Taking $x_1 = 0$ in $x_1^2 + x_2^2 - 8x_2 = 0$.

$$= 0^2 + x_2^2 - 8x_2 = 0$$

$$x_2^2 = 8x_2$$

$$x_2 = 8 \quad \text{or} \quad x_2 = 0$$

Taking $x_2 = 4$ in $x_1^2 + x_2^2 + -8x_2 = 0$

$= x_1^2 + 16 - 8(4) = 0$

$= x_1^2 + 16 - 32 = 0$

$x_1^2 = 16$

$\Rightarrow x_1 = 4$ or -4

Points : $(0,0), (0,4), (4,4), (-4,4)$

$\delta f_{x_1 x_1} = \frac{\delta^2 f}{\delta x_1^2} = 2x_1$

$\delta f_{x_2 x_2} = \frac{\delta^2 f}{\delta x_2^2} = 2x_1$

$\frac{\delta^2 f}{\delta x_1 \delta x_2} = 2x_2 - 8$

$\frac{\delta^2 f}{\delta x_2 \delta x_1}$

$H = \begin{bmatrix} \frac{\delta^2 f}{\delta x_1^2} & \frac{\delta f}{\delta x_1 \delta x_2} \\ \frac{\delta^2 f}{\delta x_2 \delta x_1} & \frac{\delta^2 f}{\delta x_2^2} \end{bmatrix}$

$H = \begin{bmatrix} 2x_1 & 2x_2 - 8 \\ 2x_2 - 8 & 2x_1 \end{bmatrix}$

H at all points :-

$H(x_0)$ at $(0,0) = \begin{bmatrix} 0 & -8 \\ -8 & 0 \end{bmatrix} = 0 - 64 = -64$ is indefinite $\rightarrow$ saddle point.

$H(x_0)$ at $(0,4) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = 0$ indefinite $\rightarrow$ saddle point.

$H(x_0)$ at $(4,4) = \begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix} = 64 - 0 = 64$ +ve definite $\rightarrow$ minimum point

$H(x_0)$ at $(-4,+4) = \begin{bmatrix} -8 & 0 \\ 0 & -8 \end{bmatrix} = 64 - 0 = 64$ -ve definite $\rightarrow$ maximum point

minimum point (4,4)
maximum point (-4,4)

2. $f(x_1, x_2, x_3) = x_1 + 2x_3 + x_2x_3 - x_1^2 - x_2^2$

$$\nabla f = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \\ \partial f/\partial x_3 \end{bmatrix} = \begin{bmatrix} 1 - 2x_1 \\ x_3 - 2x_2 \\ 2 + x_2 - 2x_3 \end{bmatrix}$$

$\nabla f = 0$

$1 - 2x_1 = 0$
$2x_1 = 1$
$x_1 = 1/2$

$x_3 - 2x_2 = 0$
$x_3 = 2x_2$

$2 + x_2 - 2x_3 = 0$
$2 + x_2 - 4x_2 = 0$
$2 - 3x_2 = 0$
$2 = 3x_2$
$x_2 = 2/3$

$x_3 = 4/3$

$\therefore x_1 = 1/2, x_2 = 2/3, x_3 = 4/3$

Hesian Matrix =

$$H(x_0) = \begin{bmatrix} f x_1x_1 & f x_1x_2 & f x_1x_3 \\ f x_2x_1 & f x_2x_2 & f x_2x_3 \\ f x_3x_1 & f x_3x_2 & f x_3x_3 \end{bmatrix}$$

$$= \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

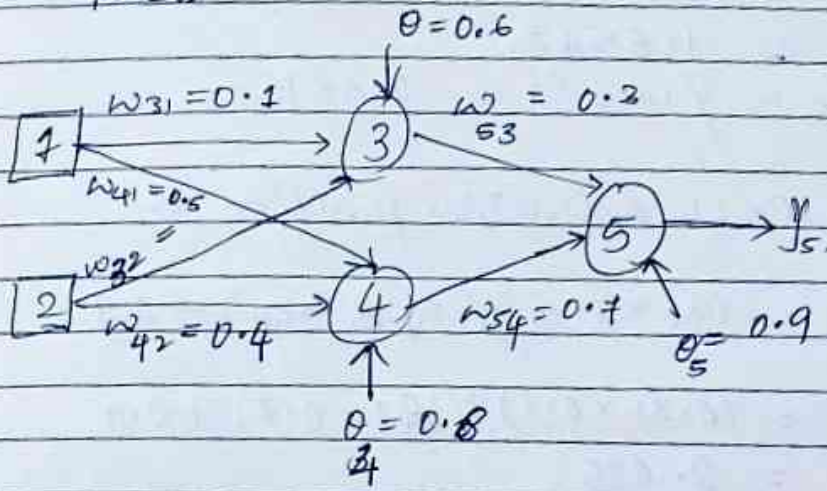
$H(x) = [-2] = -2 < 0$

$H(x) = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix} = 4 - 0 = 4$ -ve definite, maximum value.

$H(x) = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix} = -2(4-1) - 0(\ ) + 0(\ )$

$= -6 > 0$

3. Find at neuron 5, if input vector (0.7, 0.8) using the activation function ReLU.



$x_1 = 0.7 \quad x_2 = 0.3$

$H_3 = a_3 = (0.7 \times 0.1) + (0.5 \times 0.3) + 0.6 = 0.82$

$H_3 = 1/1 + e^{-a_i} = 0.6942$

$H_4 = a_4 = 1.27$

$H_4 = 1/1 + e^{-a_i} = 0.7807$

ReLU activation function

$\max(0, a)$

To find a_3, a_4

$a_3 = \max(0, 0.82) = 0.82$

$a_4 = \max(0, 1.27) = 1.27$

— By Sigmoid Activation function —

$a_5 = (0.6942)(0.3) + (0.7807)(0.7) + 0.9$

$= 1.6548$

$y_5 = 1/1 + e^{-a_5} = 0.8395.$

By ReLU activation function -

$a_5 = (a_3 \times w_{53}) + (a_4 * w_{54}) + 0.9$

$= (0.82 \times 0.3) + (1.27 \times 0.7) + 0.9$

$= 2.035$

$y_5 = \max(0, a_5) = (0, 2.035) = 2.035$

Steepest Ascent / Descent -or- Gradient Ascent / Descent

Gradient ascent / descent is an OT algorithm used in ML to minimize or maximize the cost function by iteratively adjusting parameters in the direction of the negative gradient, aiming to find the optimal set of parameters.

The cost function represents the discrepency bet^n the predicted o/p of the model and the actual o/p.

Gradient ascent / descent aims to find the parameters that minimize / maximize the discrepency and improve the model's performance.

Working rule :-

In this method, we start from an initial trial point x_0 and iteratively move along the steepest descent direction until the optimum point is found.

Step 1 :- Start with an arbitrary point x_0

Step 2 :- Determine the optimal step length λ_k [minimize $f(x + \lambda_k s_i)$; $s_i = -\nabla f$]

in the direction of $-\nabla f(x_k)$ and set $x_{k+1} = x_k - \lambda_k \nabla f(x_k)$;
$k = 0, 1, 2, 3 \ldots$.

Step 3: Test the new point x_{k+1} for optimality. If x_{k+1} is optimum, we stop the procedure. Else go to the next step.

Step 4: Set $k = k+1$ and go to step 2.

6/6/24

1. $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$
Starting from the point $x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\Rightarrow f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$

$$\nabla f = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix} = \begin{bmatrix} 1 + 4x_1 + 2x_2 \\ -1 + 2x_1 + 2x_2 \end{bmatrix} \rightarrow (1)$$

$$\nabla f(x_0) = \nabla f \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

To find optimal length (iteration 1)

$$f\left[\begin{bmatrix} 0 \\ 0 \end{bmatrix} - \lambda_0 \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right] = f\begin{bmatrix} -\lambda_0 \\ \lambda_0 \end{bmatrix}$$

$$f[-\lambda_0, \lambda_0] = -\lambda_0 - \lambda_0 + 2\lambda_0^2 - 2\lambda_0\lambda_0 + \lambda_0^2$$
$$= -2\lambda_0 + \lambda_0^2$$

$$\frac{\delta f}{\delta \lambda_0} = 2\lambda_0 - 2$$

$$\frac{df}{\delta \lambda_0} = 0 \Rightarrow 2\lambda_0 - 2 = 0$$
$$2\lambda_0 = 2$$
$$\lambda_0 = 1$$

$$x_4 = \left[\begin{bmatrix}0\\0\end{bmatrix} - 1\begin{bmatrix}1\\-1\end{bmatrix}\right] = \begin{bmatrix}-1\\1\end{bmatrix} \neq 0$$

- After substituting in (1) —

$$\nabla f(x_4) = \begin{bmatrix}-1\\-1\end{bmatrix}$$

- (iteration 2) —

minimize $f\left[x_4 - \lambda_1(\nabla f(x_4))\right]$

$$f\left[\begin{bmatrix}-1\\1\end{bmatrix} - \lambda_1\begin{bmatrix}-1\\-1\end{bmatrix}\right] = f\begin{bmatrix}-1+\lambda_1\\1+\lambda_1\end{bmatrix}$$

$$f(-1+\lambda_1, 1+\lambda_1) = -1+\lambda_1 - (1+\lambda_1) + 2(-1+\lambda_1)^2 + 2(-1+\lambda_1)(1+\lambda_1) + (1+\lambda_1)^2$$

$$= 5\lambda_1^2 - 2\lambda_1 - 1$$

$$\frac{df}{\delta \lambda_1} = 10\lambda_1 - 2 = 0 \Rightarrow \lambda_1 = 1/5$$

$$x_2 = (x_1 - \lambda_1 \nabla f(x_1))$$

$$= \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} - 1/5 \begin{bmatrix} -1 \\ -1 \end{bmatrix}\right)$$

$$= \begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

x_2 isn't an optimal point

$$\nabla f(x_2) = \begin{pmatrix} 0.2 \\ -0.2 \end{pmatrix}$$

— (iteration 3) —

minimize $f[x_2 + \lambda_2(-\nabla f(x_2))]$

$$f\left[\begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} - \lambda_2 \begin{bmatrix} 0.2 \\ -0.2 \end{bmatrix}\right] \Rightarrow f\begin{bmatrix} -0.8 - 0.2\lambda_2 \\ 1.2 + 0.2\lambda_2 \end{bmatrix}$$

$$f(-0.8 - 0.2\lambda_2, 1.2 + 0.2\lambda_2)$$

$$= (-0.8 - 0.2\lambda_2) - (1.2 + 0.2\lambda_2) + 2(-0.8 - 0.2\lambda_2)^2 + 2(-0.8 - 0.2\lambda_2)(1.2 + 0.2\lambda_2) + (1.2 + 0.2\lambda_2)^2$$

$$= -1.2 - 0.16\lambda_2 + 0.12\lambda_2^2$$

$$\partial f / \partial \lambda_2 = 0.08\lambda_2 - 0.08$$

$$= \lambda_2 = 1$$

$$x_3 = \left[x_2 - \lambda_2 \nabla f(x_2) \right]$$

$$x_3 \left[\begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} - 1 \begin{bmatrix} 0.2 \\ -0.2 \end{bmatrix} \right] = \begin{bmatrix} -1 \\ 1.4 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

x_3 is not an optimal point.

⇒ **Automatic Differentiation**

This is a special case of back propogation.

consider $f = x^2(x+y)$ where $c = x^2$, $s = x+y$

$$F \rightarrow (c, s) \rightarrow (x, y)$$

$$F \rightarrow (x, y)$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial c} * \frac{\partial c}{\partial x} + \frac{\partial f}{\partial s} * \frac{\partial s}{\partial x}$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial c} * \frac{\partial c}{\partial y} + \frac{\partial f}{\partial s} * \frac{\partial s}{\partial y}$$

$$\frac{\partial f}{\partial x} = \frac{\partial}{\partial c}[cs] * \frac{\partial}{\partial x}(x^2) + \frac{\partial}{\partial s}(cs) * \frac{\partial}{\partial x}(x+y)$$

$$\frac{\partial f}{\partial x} = (s * 2x) + (c * 1)$$

$$\boxed{\frac{\partial f}{\partial x} = 2cx + c}$$

PAGE NO :
DATE :

$$\frac{\partial f}{\partial y} = \frac{\partial}{\partial c}[cs] + \frac{\partial}{\partial y}(x^2) + \frac{\partial}{\partial s}[cs] + \frac{\partial}{\partial y}(x+y)$$

$$\frac{\partial f}{\partial y} = s * (0) + c * 1$$

$$\boxed{\frac{\partial f}{\partial y} = c}$$

⇒ Gradients in a deep network

The Function value Y is computed as a many level function composition

i.e $Y = (f_k \circ f_{k-1} \circ \cdots \circ f_1)(x)$

$$fog(x) = f(g(x))$$

Ex: $f(x) = x^2$

$g(x) = x+1$

$$f \circ g = f(g(x))$$
$$= f(x+1)$$
$$= f(x+1)^2$$
$$= (x+1)^2$$

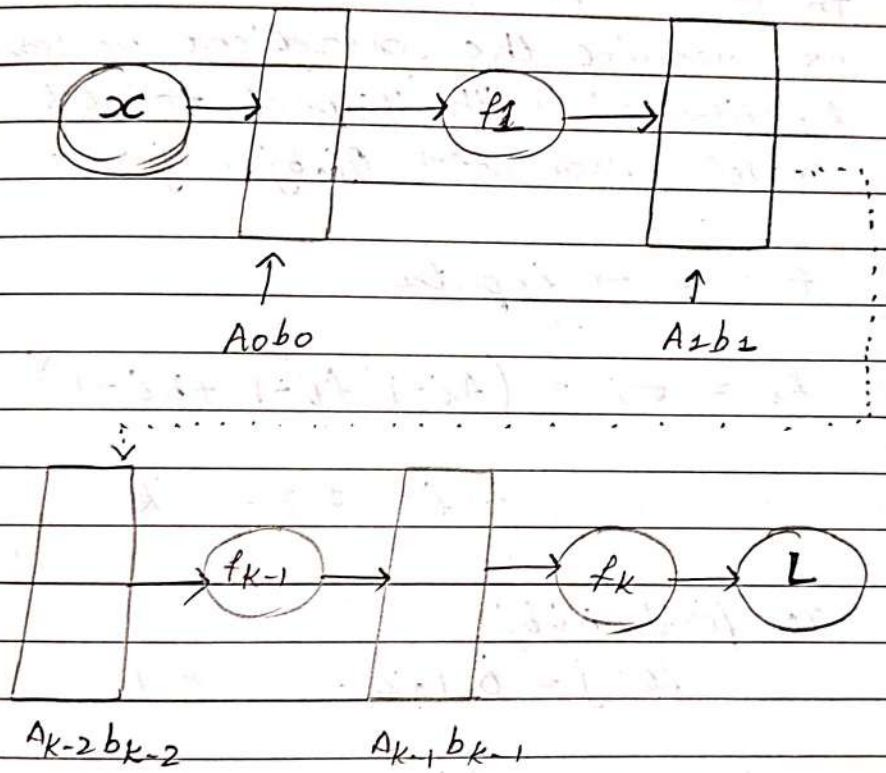
$$f_k(f_{k-1}(\cdots f_1(x))\cdots)$$

Here x are inputs or images. Y are observations

PAGE NO :
DATE :

f_i ; $i = 1, \ldots, k$, possesses its own parameters.

Forward pass is a multi-layer neural network to compute the loss 'L' as a function of the inputs x and the parameters A_i, b_i



Here $f_i(x_{i-1}) = \sigma(A_{i-1} x_{i-1} + b_{i-1})$

$f_i \rightarrow$ function is the i^{th} layer

$x_{i-1} \rightarrow$ output of layer $i-1$

$\sigma \rightarrow$ Activation function

logistic sigmoid $= \frac{1}{1 + e^{-x}}$

$$\tanh x = \frac{1 + e^{-x}}{1 + e^{x}}$$

In order to train the models we require the gradient of loss function 'L' with respect to all model parameter A_j, b_j, $j = 1 \ldots k$

$f_0 = x \rightarrow$ inputs

$$f_i = \sigma_i = (A_{i-1} f_{i-1} + b_{i-1})$$

$$= i = 1, 2 \ldots k$$

To find $A_i b_j$

for $j = 0, 1, 2 \ldots k-1$

we have to minimize the loss function

$$L(\theta) = \| y - f_k(\theta, x) \|^2$$

$$\theta = \{ A_0 \; b_0 \ldots A_{k-1}, b_{k-1} \}$$

PAGE NO :
DATE :

By definition of maxima or minima we require the partial derivative of "L" with respect to the parameter

$$\theta_j = (A_j, b_j) \Rightarrow j = 0, 1 \cdots k-1$$

By chain Rule we have

$$\frac{\partial L}{\partial \theta_{k-1}} = \frac{\partial L}{\partial f_k} \cdot \frac{\partial f_k}{\partial \theta_{k-1}}$$

$$\frac{\partial L}{\partial \theta_{k-2}} = \frac{\partial L}{\partial f_k} \cdot \frac{\partial f_k}{\partial f_{k-1}} * \frac{\partial f_{k-1}}{\partial \theta_{k-2}}$$

$\frac{\partial f_{k-1}}{\partial \theta_{k-2}}$ → partial derivative of output w.r.t parameter

$\frac{\partial f_k}{\partial f_{k-1}}$ → w.r.t inputs

Similarly

$$\frac{\partial L}{\partial \theta_{k-3}} = \frac{\partial L}{\partial f_k} * \frac{\partial f_k}{\partial \theta_{k-1}} * \frac{\partial \theta_{k-1}}{\partial \theta_{k-2}} * \frac{\partial \theta_{k-2}}{\partial \theta_{k-3}}$$

$$\vdots$$

$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial f_k} * \frac{\partial f_k}{\partial \theta_{k-1}} \cdots \cdots \frac{\partial \theta_{i+1}}{\partial \theta_i}$$

PAGE NO:
DATE:

→ Optimization using second derivative test

Procedure

1. A necessary condition for x_0 to be an extreme point (maximum or minimum) of $f(x)$ is $\nabla f(x_0) = 0$ where $x \in R^n$ and $x_0 \in R^n$ is called critical point or stationary point.

2. A sufficient condition for a stationary point x_0 of $f(x)$ to be an extreme (maximum or minimum) is that

case-1: f has relative minimum if

$$f_{xx}(x_0) f_{yy}(x_0) - f_{xy}^2(x_0) > 0$$

and

$$f_{xx}(x_0) > 0$$

case-2: f has relative maximum if

$$f_{xx}(x_0) f_{yy}(x_0) - f_{xy}^2(x_0) > 0$$

and

$$f_{xx}(x_0) < 0$$

case-3: f has a saddle point (neither maximum or minimum) if

$$f_{xx}(x_0) f_{yy}(x_0) - f_{xy}^2(x_0) < 0$$

case-4: The test is inconclusive if

$$f_{xx}(x_0) f_{yy}(x_0) - f_{xy}^2(x_0) = 0$$

PAGE NO :
DATE :

→ Hessian matrix form

A sufficient condition for a stationary point x_0 of $f(x)$ to be extremum (maximum or minimum)

case : 1 : f has relative minimum if $f(h_0)$ is positive definite

case : 2 - f has relative maximum if $f(h_0)$ is negative definite

case : 3 - f has a saddle point if $f(h_0)$ is indefinite

case : 4 - The test is inconclusive otherwise

Problems

1. Using the Hessian matrix to classify the relative extremum for $f(x_1, x_2) = \frac{1}{3}x_1^3 + x_1x_2^2 - 8x_1x_2 + 3$

solution $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2^2 - 8x_2 \\ 2x_2x_1 - 8x_1 \end{bmatrix}$

Equate to zero.

$x_1^2 + x_2^2 - 8x_2 = 0 \quad -(1)$

$2x_2x_1 - 8x_1 = 0$

DATE :

$2x_2x_1 = 8x_1$

$2x_2 = \frac{8x_1}{x_1}$

$2x_2 = 8$

$x_2 = \frac{8}{2}$

$\boxed{x_2 = 4}$

$x_1^2 + x_2^2 - 8x_2 = 0$

$x_1^2 + (4)^2 - 8(4) = 0$

$x_1^2 + 16 - 32 = 0$

$x_1^2 - 16 = 0$

$x_1^2 = 16$

$x_1 = \pm 4$

with $x_2 = 4$

Stationary point = $(+4, 4), (-4, 4)$

$(0, 0), (0, 8)$

PAGE NO :
DATE :

$2x_1x_2 - 8x_1 = 0$ ... (1)

$2x_1(x_2 - 4) = 0$

$2x_1 = 0$

$x_2 - 4 = 0$

$x_1 = 0, \; x_2 = 4.$

$$f_{x_1x_1} = \frac{\partial^2 f}{\partial x_1^2} = \frac{\partial}{\partial x_1}(x_1^2 + x_2^2 - 8x_2)$$

$$= 2x_1$$

$$f_{x_2x_2} = \frac{\partial^2 f}{\partial x_2^2} = \frac{\partial}{\partial x_2}(2x_2x_1 - 8x_1)$$

$$= 2x_1.$$

$$f_{x_1x_2} = \frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_1}(2x_2x_1 - 8x_1)$$

$$= 2x_2 - 8$$

$$f_{x_2x_1} = \frac{\partial^2 f}{\partial x_2 \partial x_1} = \frac{\partial}{\partial x_2}(x_1^2 + x_2^2 - 8x_2)$$

$$= 2x_2 - 8$$

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$$

PAGE NO :
DATE :

$$H = \begin{bmatrix} 2x_1 & 2x_2 - 8 \\ 2x_2 - 8 & 2x_1 \end{bmatrix}$$

H at (0,0)

$$H = \begin{bmatrix} 0 & -8 \\ -8 & 0 \end{bmatrix}$$

$= 0 - (-64) = -64$

saddle point

H at (0, 8)

$$H = \begin{bmatrix} 0 & 8 \\ 8 & 0 \end{bmatrix}$$

$0 - (64) = -64$

indefinite

saddle point

H at (4, 4)

$$H = \begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}$$

$64 - 0 = 64$

minimum point

H at (-4, 4)

$$H = \begin{bmatrix} -8 & 0 \\ 0 & -8 \end{bmatrix}$$

$64 - 0 = 64$

maximum point

2.

solution

PAGE NO :
DATE :

2. Using the Hessian matrix to classify the relative extremum

$f(x_1, x_2, x_3) = x_1 + 2x_3 + x_2x_3 - x_1^2 - x_2^2 - x_3^2$

Solution

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 1 - 2x_1 \\ x_3 - 2x_2 \\ 2 + x_2 - 2x_3 \end{bmatrix}$$

$\nabla f = 0$

$1 - 2x_1 = 0 \quad -(1)$

$x_3 - 2x_2 = 0 \quad -(2)$

$2 + x_2 - 2x_3 = 0 \quad -(3)$

$1 - 2x_1 = 0$

$1 = 2x_1$

$\boxed{\frac{1}{2} = x_1}$

$\boxed{x_3 = 2x_2}$

$\frac{x_3}{x_2} = 2$

$2 + x_2 - 2(2x_2) = 0$

$2 + x_2 - 4x_2 = 0$

PAGE NO :
DATE :

$$2 - 3x_2 = 0$$

$$2 = 3x_2$$

$$\boxed{\frac{2}{3} = x_2}$$

$$x_3 - 2x_2 = 0$$

$$x_3 - 2\left(\frac{2}{3}\right) = 0$$

$$x_3 - \frac{4}{3} = 0$$

$$\boxed{x_3 = \frac{3}{4}}$$

$$(x_1, x_2, x_3) = (1/2, 2/3, 3/4)$$

$$f_{x_1x_1} = \frac{\partial^2 f}{\partial x_1^2} = \frac{\partial}{\partial x_1}(1 - 2x_1) = -2$$

$$f_{x_2x_2} = \frac{\partial^2 f}{\partial x_2 \partial x_2} = \frac{\partial}{\partial x_2}(x_3 - 2x_2) = -2$$

$$f_{x_3x_3} = \frac{\partial^2 f}{\partial x_3 \partial x_3} = \frac{\partial}{\partial x_3}(2 + x_2 - 2x_3) = -2$$

$$f_{x_1x_2} = f_{x_2x_1} = \frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_1}(x_3 - 2x_2) = 0$$

$$f_{x_1x_3} = f_{x_3x_1} = \frac{\partial^2 f}{\partial x_1 \partial x_3} = \frac{\partial}{\partial x_1}(2 + x_2 - 3x_3) = 0$$

DATE :

$$f_{x_2x_3} = f_{x_3x_2} = \frac{\partial^2 f}{\partial x_2 \partial x_3} = \frac{\partial}{\partial x_2}(2 + x_2 \cdot 2x_3)$$

$$= 1.$$

$$H(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \frac{\partial^2 f}{\partial x_1 x_3} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 x_3} \\ \frac{\partial^2 f}{\partial x_3 x_1} & \frac{\partial^2 f}{\partial x_3 x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{bmatrix}$$

$$H(x) = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

$$H(x) = -2(x - 1) + 0 + 0$$

$$H(x) = -2(3)$$

$$= -6$$

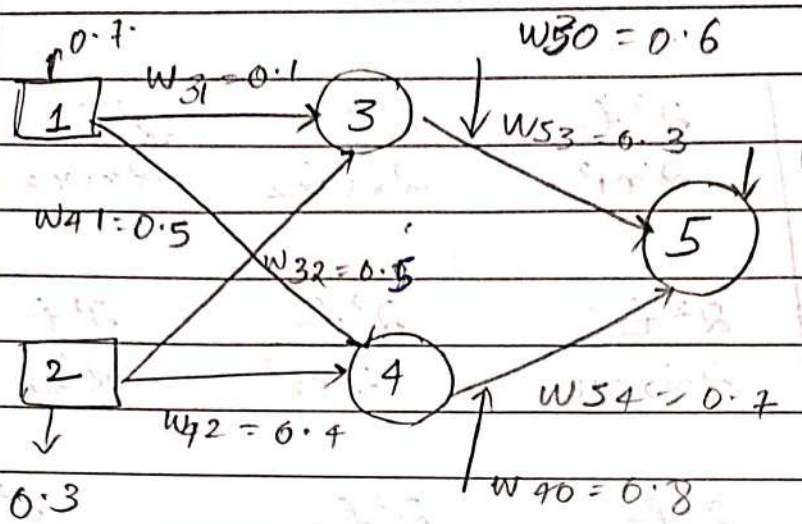
$$H(x) = -2$$

$$H(x) = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix} = 4 > 0$$

$$H(x) = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix} = -6 > 0$$

DATE :

Find output at neuron 5 if input vector [0.7, 0.3] using the activation function ReLu



Logistic sigmoid function = $\frac{1}{1+e^{-a_i}}$

$$a_3 = (w_{31} * 0.7) + (w_{32} * 0.3) + w_{30}$$
$$= (0.1 * 0.7) + (0.5 * 0.3) + 0.6$$
$$= 0.07 + 0.15 + 0.6$$
$$= 0.82$$

$$y_3 = F(a_3)$$
$$= \frac{1}{1+e^{-0.82}}$$

$$y_3 = 0.6942$$

PAGE NO :
DATE :

$a_4 = (w_{41} * 0.7) + (w_{42} * 0.3) + w_{40}$

$a_4 = (0.5 * 0.7) + (0.4 * 0.3) + 0.8$

$a_4 = 0.35 + 0.12 + 0.8$

$a_4 = 1.27$

$y_4 = F(a_4)$

$= \frac{1}{1 + e^{-a_i}}$

$= \frac{1}{1 + e^{-1.27}}$

$y_4 = 0.781$

ReLu Activation Function

$\max(0, a)$

To find a_3

$\max(0, a_3)$

$= \max(0, 0.82)$

$a_3 = 0.82 = y_3$

To find a_4

$\max(0, a_4)$

$= \max(0, 1.27)$

$a_4 = 1.27 = y_4$

$a_5 = (y_3 * w_{53}) + (y_4 * w_{54}) + w_{50}$

~~$= (0.6942 * 0.3) + ($~~

$= (0.82 * 0.3) + (1.27 * 0.7) + 0.9$

$= 0.2460 + 0.8890 + 0.9 = 2.0350$

PAGE NO :
DATE :

$y_5 = \max(0, 95)$

$y_5 = \max(0, 2.035)$

$\boxed{y_5 = 2.035}$

→ Steepest Ascent / Descent

(OR)

Gradient Ascent / Descent method

Gradient descent is an optimization algorithm used in machine learning to minimize or maximum the cost function by iteratively adjusting parameters in the direction of negative gradient aiming to find the optimal set of parameter

The cost function represents the discrepency between predicted output of the model and the actual output.

Gradient Descent or accent aims to find the parameters that minimize or maximize this discrepency and improve the model performance.

PAGE NO :
DATE :

Working Principle

In this method we start from a initial trial point x_0 and iteratively move along the steepest descent direction until the optimum point is found.

1. Start with an arbitary point x_0

2. Determine the optimal step length λ_k [minimize ~~or maximum~~) $f(x + \lambda_k s_i \cdot s_i = -\nabla f$ in the direction of $-\nabla f(x_k)$ and

   set $x_{k+1} = x_k - \lambda_k \nabla f(x_k)$; $k = 0, 1, 2 \ldots$

3. Test the new point x_{k+1} for optimoulity. If x_{k+1} is optimum we stop the process otherwise go to the next step.

4. Set $k = k+1$ and go to step 2

PAGE NO :
DATE :

Problems

1. $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$

Starting from the point $x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Solution: $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$

$$\nabla f = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix} = \begin{bmatrix} 1 + 4x_1 + 2x_2 & \text{①} \\ -1 + 2x_2 + 2x_2 & \text{②} \end{bmatrix}$$

$$\nabla f(x_0) = \nabla f \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 + 4(0) + 2(0) \\ -1 + 2(0) + 2(0) \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ -1 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

To find optimal length

minimize $f[x_0 + \lambda_0(-\nabla f(x_0))]$

$$f\left[\begin{bmatrix} 0 \\ 0 \end{bmatrix} + -\lambda_0 \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right]$$

$$f = \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix} - \lambda_0 \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right]$$

PAGE NO :
DATE :

$$f = \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -\lambda_0 \\ \lambda_0 \end{bmatrix} \right]$$

$$f = \begin{bmatrix} -\lambda_0 \\ \lambda_0 \end{bmatrix}$$

Substitute in $f(x_1, x_2)$.

$$f(-\lambda_0, \lambda_0) = -\lambda_0 - \lambda_0 + 2\lambda_0 + 2\lambda_0\lambda_0 + \lambda_0^2$$

$$f(-\lambda_0, \lambda_0) = -2\lambda_0 + \lambda_0^2$$

$$\frac{2f}{2\lambda_0} = 2\lambda_0 - 2$$

$$\frac{2f}{2\lambda_0} = 0$$

$$2\lambda_0 - 2 = 0$$

$$2\lambda_0 = 2$$

$$\lambda_0 = 1$$

$$x_1 = \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix} - 1 \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right]$$

$$x_1 = \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right]$$

$$x_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Substitute in (1) and (2)

PAGE NO :
DATE :

$$\nabla f(x_1) = \begin{bmatrix} 1 + 1(-1) + 2(1) \\ -1 + 2(-1) + 2(1) \end{bmatrix}$$

$$\nabla f(x_1) = \begin{bmatrix} 1 - 1 + 2 \\ -1 - 2 + 2 \end{bmatrix}$$

$$\nabla f(x_1) = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

Iteration 2

Minimize $f\left[x_1 - \lambda_1 (\nabla f(x_1))\right]$

$$f\left[\begin{bmatrix} -1 \\ 1 \end{bmatrix} - \lambda_1 \begin{bmatrix} 2 \\ -1 \end{bmatrix}\right]$$

$$f\left[\begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} \lambda_1 \\ \lambda_1 \end{bmatrix}\right]$$

$$f\begin{bmatrix} -1 + \lambda_1 \\ 1 + \lambda_1 \end{bmatrix}$$

PAGE NO :
DATE :

$f(-1+\lambda_1, 1+\lambda_1) = -1+\lambda_1 - 1-\lambda_1 + 2(-1+\lambda_1)^2 + 2(-1+\lambda_1)(1+\lambda_1) + (1+\lambda_1)^2$

$= -2 + 2(\lambda_1^2 + 1 - 2\lambda_1) + 2(-1 + \lambda_1(1+\lambda_1)) + 1 + \lambda_1^2 + 2\lambda_1$

~~$-2 + 2(\lambda_1^2 + 1 - 2\lambda_1)$~~

$= -2 + 2\lambda_1^2 + 2 - 4\lambda_1 + 1 + \lambda_1^2 + 2\lambda_1 + 2(-1 - \lambda_1 + \lambda_1 + \lambda_1^2)$

$= 3\lambda_1^2 - 2\lambda_1 - 2 - 2\lambda_1 + 2\lambda_1 + 1 + 2\lambda_1^2$

$= 5\lambda_1^2 - 2\lambda_1 - 1$

$\frac{2f}{2\lambda_1} = 10\lambda_1 - 2$

$\frac{2f}{2\lambda_1} = 0$

$10\lambda_1 - 2 = 0$

$10\lambda_1 = 2$

$\lambda_1 = \frac{2}{10}$

$\lambda_1 = \frac{1}{5}$

$\lambda_1 = 1/5$

$x_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \frac{1}{5}\begin{bmatrix} -1 \\ -1 \end{bmatrix}$

DATE :

$$x_2 = \left[\begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1/5 \\ 1/5 \end{bmatrix} \right]$$

$$x_2 = \begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

x_2 is not optimal

Substitute x_2 value in ① and ②

$$\nabla f(x_2) = \begin{bmatrix} 1 + 4(-0.8) + 2(1.2) \\ -1 + 2(-0.8) + 2(1.2) \end{bmatrix}$$

$$\nabla f(x_2) = \begin{bmatrix} 1 - 3.2 + 2.4 \\ -1 - 1.6 + 2.4 \end{bmatrix}$$

$$\nabla f(x_2) = \begin{bmatrix} 0.2 \\ -0.2 \end{bmatrix}$$

Iteration 3.

Minimize $f\left[x_2 + \lambda_2 (-\nabla f(x_2)) \right]$

$$f\left[\begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} - \lambda_2 \begin{bmatrix} 0.2 \\ -0.2 \end{bmatrix} \right]$$

PAGE NO :
DATE :

$$f\begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} + \begin{bmatrix} -\lambda_2 0.2 \\ +\lambda_2 0.2 \end{bmatrix}$$

$$f\begin{bmatrix} -0.8 - \lambda_2 0.2 \\ 1.2 + \lambda_2 0.2 \end{bmatrix}$$

$$f(-0.8 - \lambda_2 0.2,\ 1.2 + \lambda_2 0.2)$$

Substitute in main equation.

$$= (-0.8 - \lambda_2 0.2) - (1.2 + \lambda_2 0.2) + 2(-0.8 - \lambda_2 0.2)^2 + 2(-0.8 - \lambda_2 0.2(1.2 + \lambda_2 0.2)) + (1.2 + \lambda_2 0.2)^2$$

$$= (-0.8 - \lambda_2 0.2 - 1.2 - \lambda_2 0.2) + 2(0.64 + 0.04\lambda_2^2 - 0.32\lambda_2) + 2(-0.96 - 0.16\lambda_2 - 0.24\lambda_2 - 0.2\lambda_2^2) + (1.44 + 0.04\lambda_2 0.04\lambda_2^2 + 0.48\lambda_2)$$

$$= (-0.8) - \lambda_2 0.2 - (1.2) - \lambda_2 0.2 + (1.28) + 0.08\lambda_2^2 - 0.64\lambda_2 - (1.92) - 0.32\lambda_2 - 0.48\lambda_2 - 0.4\lambda_2^2 + (1.44) + 0.04\lambda_2^2 + 0.48\lambda_2$$

$$= -1.2 - 1.36\lambda_2 - 0.28\lambda_2^2$$

$$\frac{\partial f}{\partial x_1} = -1.36 - 0.56\lambda_2$$

$$-1.36 = 0.56\lambda_2$$

PAGE NO :
DATE :

$$(0.8 - 0.2\lambda_2) \div (1.2 + 0.2\lambda_2)$$
$$-2(0.8 + 0.2\lambda_2)^2$$
$$+2(-0.8 - 0.2\lambda_2)(1.2 + \lambda_2 0.2)$$
$$+ (1.2 + 0.2\lambda_2)^2$$

$$\frac{2f}{2\lambda_2} = 0.08\lambda_2 - 0.08$$

$$\lambda_2 = 1$$

$$X_3 = (X_2 - \lambda_2 \nabla f(x_2))$$

$$X_3 = \left[\begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} - 1\begin{bmatrix} 0.2 \\ -0.2 \end{bmatrix}\right]$$

$$= \begin{bmatrix} -1 \\ 1.4 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

X_3 is not optimal point.

2. $f(x, y) = 4x^2 - 4xy + 2y^2$
$x^{(0)}, y^{(0)} = (2, 3)$

tion $f(x, y) = 4x^2 - 4xy + 2y^2$

$$\nabla f = \begin{bmatrix} 2f/2x \\ 2f/2y \end{bmatrix} = \begin{bmatrix} 8x - 4y \\ -4x + 4y \end{bmatrix}$$

PAGE NO :
DATE :

Substituting $(x, y) = (2, 3)$

$$\nabla f(x_0) = \begin{bmatrix} (8*2) - 1(3) \\ -1(2) + 1(3) \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix} \neq \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

(Iteration 1)

Minimize $f[x_0 + \lambda_0(-\nabla f(x_0))]$

$$= f\left[\begin{bmatrix} 2 \\ 3 \end{bmatrix} - \lambda_0 \begin{bmatrix} 4 \\ 1 \end{bmatrix}\right]$$

$$= f\begin{bmatrix} 2 - 4\lambda_0 \\ 3 - 4\lambda_0 \end{bmatrix}$$

$$= f(2 - 4\lambda_0, 3 - 4\lambda_0)$$

$$= 4(2 - 4\lambda_0)^2 - 4(2 - 4\lambda_0)(3 - 4\lambda_0) + 4(3 - 4\lambda_0)^2$$

$$= 4(4 + 16\lambda_0^2 - 16\lambda_0) - 4(6 - 8\lambda_0 - 12\lambda_0 + 16\lambda_0^2) + 2(9 + 16\lambda_0^2 - 24\lambda_0)$$

$$= 10 + 32\lambda_0^2 - 32\lambda_0$$

$$\frac{2f}{2\lambda_0} = 64\lambda_0 - 32$$

equate to 0

$$\frac{2f}{2\lambda_0} = 0$$

$$64\lambda_0 - 32 = 0$$

$$64\lambda_0 = 32$$

$$\lambda_0 = \frac{32}{64}$$

$$\lambda_0 = 0.5$$

$$x_1 = x_0 + \lambda_1 (-\nabla f(x_0))$$

$$= \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \lambda_1 \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \neq \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$\nabla f(x_1) = \begin{bmatrix} 8(0) - 4(1) \\ -4(0) + 4(1) \end{bmatrix}$$

$$= \begin{bmatrix} -4 \\ +4 \end{bmatrix}$$

Iteration (2) ——

PAGE NO :
DATE :

~~minimize f(x1 - ∇λ1~~

minimize $f(x_1 + \lambda_1(-\nabla f(x_1))$

$$= f\left[\begin{bmatrix}0\\1\end{bmatrix} - \lambda_1\begin{bmatrix}-4\\4\end{bmatrix}\right]$$

$$= f\left[\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}4\lambda_1\\-4\lambda_1\end{bmatrix}\right]$$

$$= \begin{bmatrix}4\lambda_1\\1-4\lambda_1\end{bmatrix}$$

$f(4\lambda_1, 1-4\lambda_1)$

$= 4(4\lambda_1)^2 - 4(4\lambda_1)(1-4\lambda_1) + 2(1-4\lambda_1)^2$

$= 4(16\lambda_1^2) - 4(4\lambda_1 - 16\lambda_1^2) + 2(1 + 16\lambda_1^2 - 8\lambda_1)$

$= 64\lambda_1^2 - 16\lambda_1 + 64\lambda_1^2 + 2 + 32\lambda_1^2 - 16\lambda_1$

$= 160\lambda_1^2 - 32\lambda_1 + 2$

$\frac{df}{d\lambda_1} = 320\lambda_1 - 32$

equate it to zero.

PAGE NO :
DATE :

$$\frac{2f}{2\lambda_1} = 0$$

$$320\lambda_1 - 32 = 0$$

$$320\lambda_1 = 32$$

$$\lambda_1 = \frac{32}{320}$$

$$\boxed{\lambda_1 = 1/10}$$

$$x_2 = x_1 - \lambda_1 \nabla f(x_1)$$

$$x_2 = \left[\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \frac{1}{10}\begin{bmatrix} -4 \\ 4 \end{bmatrix}\right]$$

$$x_2 = \begin{bmatrix} 2/5 \\ 3/5 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

x_2 is not optimal

PAGE NO :
DATE :

Problems On Automatic Differentiation

1. Draw a computational graph of the function $f(x) = \sqrt{x^2 + e^{x^2} + \cos(x^2 + e^{x^2})}$ also find $\frac{df}{dx}$ by using automatic differentiation.

Solution Let $x^2 = a$

$e^{x^2} = e^a = b$

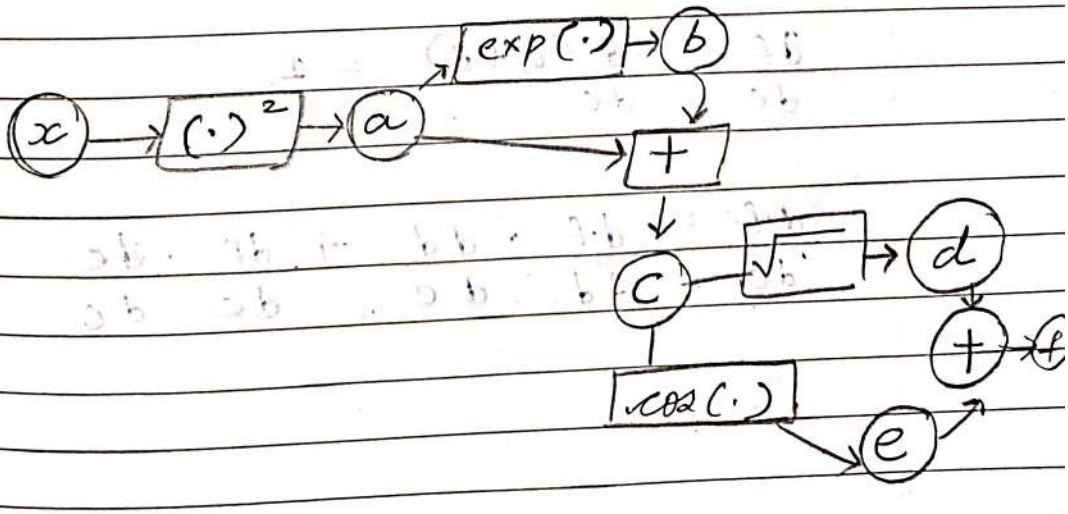
$c = a + b$

$d = \sqrt{c}$

$e = \cos(c)$

$f(x) = d + e$

computational Graph



PAGE NO :
DATE :

To find $\frac{df}{dx}$

$$\frac{da}{dx} = \frac{d(x^2)}{dx} = 2x$$

$$\frac{db}{da} = \frac{d}{da}(e^a) = e^a$$

$$\frac{dc}{da} = \frac{d}{da}(a+b) = 1$$

$$\frac{dc}{db} = \frac{d}{db}(a+b) = 1$$

$$\frac{dd}{dc} = \frac{d}{dc}\sqrt{c} = \frac{1}{2\sqrt{c}}$$

$$\frac{de}{dc} = \frac{d}{dc}(\cos c) = -\sin(c)$$

$$\frac{df}{dd} = \frac{d}{dd}(d+e) = 1$$

$$\frac{df}{de} = \frac{d}{de}(d+e) = 1$$

$$\frac{df}{dc} = \frac{df}{dd} \cdot \frac{dd}{dc} + \frac{df}{de} \cdot \frac{de}{dc}$$

PAGE NO :
DATE :

$$\frac{df}{dc} = 1\left[\frac{1}{2\sqrt{c}}\right] + 1\left[-\sin(c)\right]$$

$$\frac{df}{db} = \frac{df}{dc}\cdot\frac{dc}{dd} =$$

$$\frac{df}{db} = \frac{df}{dc}$$

$$\frac{df}{da} = \frac{df}{db}\cdot\frac{db}{da} + \frac{df}{dc}\cdot\frac{dc}{da}$$

$$= \frac{df}{db}\cdot e^a + \frac{df}{dc}(1)$$

$$= \frac{df}{dc}\left[e^a + 1\right]$$

$$\frac{df}{dx} = \frac{df}{da}\cdot\frac{da}{dx}$$

$$= \frac{df}{da}(2x)$$

$$= \frac{df}{dc}\cdot(e^a+1)(2x)$$

$$= \left[\frac{1}{2\sqrt{c}}\right] - \left[\sin(c)\right]\left[e^a+1\right]\left[2x\right]$$

PAGE NO :
DATE :

(2) Let $f(x_1, x_2) = \log x_1 + x_1 x_2 - \sin x_2$

1. Draw a computational graph of $f(x_1, x_2)$

2. Evaluate f at $x_1, x_2 = (2, 5)$ by Forward trace

Solution

$x_1 = v_0$

$x_2 = v_1$

$v_3 = \log(v_0) = \log(2) = 0.693$ ($\ln(2)$)

$v_4 = v_0 * v_1$

$= x_1 * x_2$

$= 2 * 5$

$v_4 = 10$

$\rightarrow v_5 = v_3 + v_4 = 10.693$

$v_6 = -\sin(v_1)$

$= -\sin(5)$

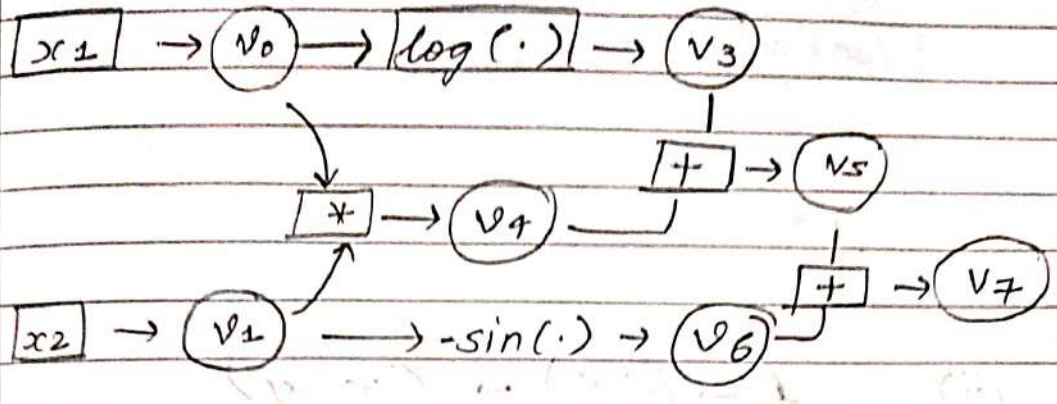
$= 0.9589$

$v_7 = v_5 + v_6$

$v_7 = 10.693 + 0.9589$

$v_7 = 11.643$

PAGE NO :
DATE :



3. If $f = x^2 (x+y)$ where $c = x^2$, $s = x+y$

(i) Draw a computational graph.

(ii) $\frac{df}{dx}$, $\frac{df}{dy}$ using chain Rule

(iii) Draw a computational graph in forward mode to show the results of (ii)

Solution

$x = a$

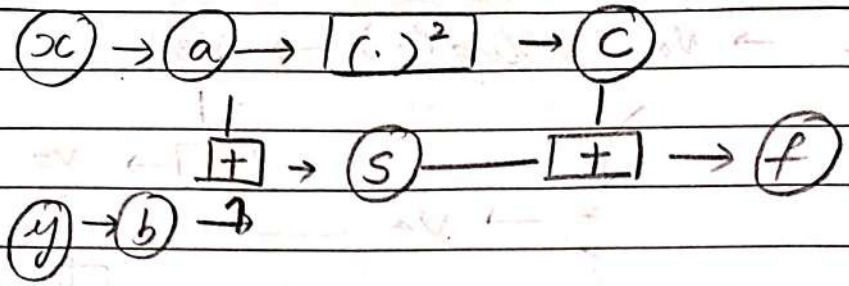
$y = b$

$c = x^2 = a^2$

$s = a+b = x+y$

$f = c(s)$

DATE :



(ii) $f \to (c\ s) \to (x\ y)$

$f \to (x\ y)$

$$\frac{df}{dx} = \frac{df}{dc} \cdot \frac{dc}{dx} + \frac{df}{ds} \cdot \frac{ds}{dx}$$

$$\frac{df}{dx} = \frac{d(cs)}{dc} \cdot \frac{d(x^2)}{dx} + \frac{d(cs)}{ds} \cdot \frac{d(x+y)}{dx}$$

$$\frac{df}{dx} = s(2x) + c(\cdot 1)$$

$$\frac{df}{dx} = s(2x) + c$$

$$\frac{df}{dx} = (x+y)2x + x^2$$

$$\frac{df}{dx} = 2x^2 + 2xy + x^2$$

$$\frac{df}{dx} = 3x^2 + 2xy$$

PAGE NO:
DATE:

$$\frac{df}{dy} = \frac{df}{dc} \cdot \frac{dc}{dy} + \frac{df}{ds} \cdot \frac{ds}{dy}$$

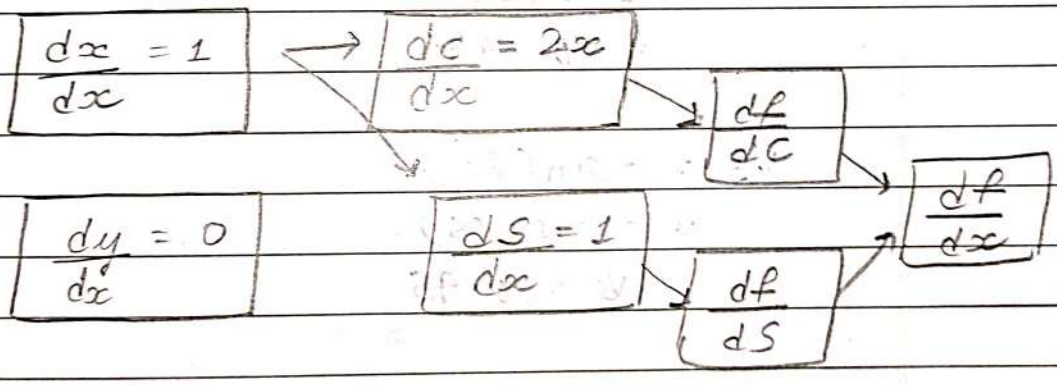
$$\frac{df}{dy} = \frac{d}{dc}(cs) \cdot \frac{d}{dy}(x^2) + \frac{d}{ds}(cs) \cdot \frac{d(x+y)}{dy}$$

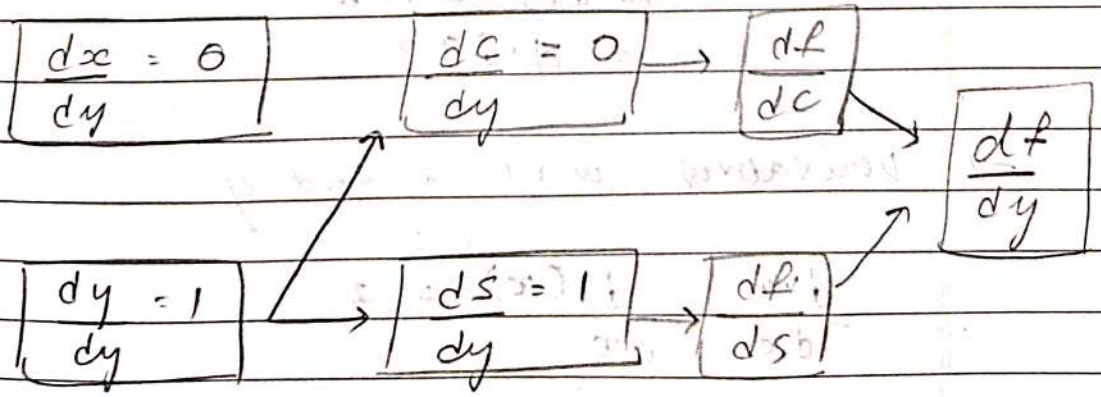
$$\frac{df}{dy} = s(0) + c \cdot 1$$

$$\frac{df}{dy} = c = x^2$$

iii) Computational graph to show results

a) Graph of x and y





DATE :

4. If $f(x,y) = \log x + xy - \sin y$ compute the forward trace derivative of f w.r.t x, and y at $(2,5)$

Solution

$x = v_0 = 2$

$y = v_1 = 5$

$v_2 = \log(v_0)$

$= \log(2)$

$v_2 = 0.6931$

$v_3 = v_0 * v_1$

$= 2 * 5$

$v_3 = 10$

$v_4 = v_2 + v_3$

$= 0.6931 + 10$

$= 10.693$

$v_5 = -\sin(v_1)$

$v_5 = -\sin(5)$

$v_5 = 0.95$

$v_6 = v_4 + v_5$

$= 10.693 + 0.958$

$= 11.643$

→ Derivatives w.r.t x and y

$\frac{dv_0}{dx} = \frac{d}{dx}(x) = 1$

PAGE NO :
DATE :

$$\frac{dv_1}{dx} = \frac{d}{dx}(4) = 0$$

$$\frac{dv_2}{dx} = \frac{d}{dx}(\log v_0) = \frac{d}{dx}(1.\log x) = \frac{1}{x} = \frac{1}{2} = 0.5$$

$$\frac{dv_3}{dx} = \frac{d}{dx}(v_0 v_1)$$

$$= v_0 \cdot \frac{d}{dx}(v_1) + v_1 \cdot \frac{d}{dx}(v_0)$$

$$= v_0(0) + v_1(1)$$

$$= 2(0) + 5(1)$$

$$= 5$$

$$\frac{dv_4}{dx} = \frac{d}{dx}(v_2 + v_3)$$

$$= \frac{dv_2}{dx} + \frac{dv_3}{dx}$$

$$= 0.5 + 5$$

$$= 5.5$$

$$\frac{dv_5}{dx} = \frac{d}{dx}(-\sin(v_1))$$

$$= -\cos v_1 \cdot \frac{dv_1}{dx}$$

$$= -\cos(v_1) \cdot 0$$

$$= 0$$

PAGE NO:
DATE:

$$\frac{dv_6}{dx} = \frac{d}{dx}(v_4 + v_5)$$

$$= \frac{dv_4}{dx} + \frac{dv_5}{dx}$$

$$= 5 \cdot 5 + 0$$

$$= 5 \cdot 5$$

Derivative with respect to y.

$$\frac{dv_0}{dy} = \frac{d}{dy}(x) = 0$$

$$\frac{dv_1}{dy} = \frac{d}{dy}(y) = 1$$

$$\frac{dv_2}{dy} = \frac{d}{dy}(\log v_0)$$

$$= \frac{1}{\log(v_0)} \cdot \frac{dv_0}{dy}$$

$$= \frac{1}{\log 2} \cdot 0$$

$$\frac{dv_2}{dy} = 0.$$

PAGE NO :
DATE :

$$\frac{dv_3}{dy} = \frac{d}{dy}(v_0 \cdot v_1)$$

$$= v_0 \cdot \frac{dv_1}{dy} + v_1 \cdot \frac{dv_0}{dy}$$

$$= v_0 \cdot (1) + v_1 (0)$$

$$= 2(1) + 5(0)$$

$$= 2$$

$$\frac{dv_4}{dy} = \frac{d}{dy}(v_2 + v_3)$$

$$= \frac{dv_2}{dy} + \frac{dv_3}{dy}$$

$$= 0 + 2$$

$$= 2$$

$$\frac{dv_5}{dy} = \frac{d}{dy}(-\sin(v_1))$$

$$= -\cos(v_1) \cdot \frac{d(v_1)}{dy}$$

$$= -\cos(5) \cdot 1$$

$$= -0.2837$$

$$\frac{dv_6}{dy} = \frac{d}{dy}(v_4 + v_5)$$

$$= \frac{dv_4}{dy} + \frac{dv_5}{dy} = 2 - 0.2837$$

$$= 1.7163$$

PAGE NO :
DATE :

Gradients of Quadratic cost

obtain the quadratic cost of function

$$C = (\hat{y} - y)^2$$

$$C = (\hat{y} - y)^2$$

$$u = \hat{y} - y$$

$$C = u^2$$

$\hat{y}$ → Estimated value

y → True value

$$\frac{dC}{du} = 2u = 2(\hat{y} - y)$$

$$u = \hat{y} - y$$

$$\frac{du}{d\hat{y}} = 1$$

$$\frac{dC}{d\hat{y}} = \frac{dC}{du} \cdot \frac{dC}{d\hat{y}}$$

$$= 2(\hat{y} - y) \cdot 1$$

Estimated y w.r.t model parameters

$$y = mx + b$$

PAGE NO :
DATE :

$$\frac{d\hat{y}}{dm} = \frac{d}{dm}(mx+b)$$

$$= x$$

$$\frac{d\hat{y}}{db} = \frac{d}{db}(mx+b)$$

$$= 1$$

Quadratic cost w.r.t model parameters

$$\frac{dC}{db} = \frac{dC}{d\hat{y}} \cdot \frac{d\hat{y}}{db}$$

$$\frac{dC}{dm} = \frac{dC}{d\hat{y}} \cdot \frac{d\hat{y}}{dm}$$

$$\nabla C = \begin{bmatrix} \frac{dC}{db} \\ \\ \frac{dC}{dm} \end{bmatrix} = \begin{bmatrix} 2(\hat{y} - y) \\ \\ 2(\hat{y} - y)(x) \end{bmatrix}$$

PAGE NO :
DATE :

2. Obtain the gradient of mean squared error [average quadratic cost]

Solution

mean Estimated Error

$$(C) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y_i} - y_i)^2$$

$$u = \hat{y}i - y_i$$

$$C = \frac{1}{n} \sum u^2$$

$\hat{y_i}$ → Estimated value

y_i → True value

$$\frac{dC}{du} = \frac{1}{n} \sum 2u$$

$$= \frac{1}{n} \sum 2(\hat{y_i} - y_i)$$

$$u = \hat{y_i} - y_i$$

PAGE NO :
DATE :

=> Fibonacci Sequence

1 1 2 3 5 8 13 21 34 55
F_0 F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9

$F_0 = 1 = F_1$, $F_n = F_{n-1} + F_{n-2}$, $n \geq 2$

→ Fibonacci Search method

Algorithm / Method

Step 1. Determine the number of iterations

Step 2. Define $L_2^* = \frac{F_{n-2}}{F_n} * L_0$

where $L_0 = b - a$

$\Rightarrow L_2^* = \frac{F_{n-2}}{F_n}(b-a)$ → Length of interval uncertainty

choose the two initial points x_1 and x_2 be such that $x_1 = a + L_2^*$

$\Rightarrow \frac{a + F_{n-2} * L_0}{F_n}$

$x_2 = a - L_2^* = b - \frac{F_{n-2} * L_0}{F_n}$

Step: 3 Compute $F(x_1)$ and $F(x_2)$ and apply unimodel principle

PAGE NO :
DATE :

Process is repeated again and again

Unimodel Principle

Let initial interval $[a, b]$ be given and the number of iterations be n

1. Set $L = a$ (Left value), $R = b$ (Right value), $k = 1$ (iteration number)

2. Compute x_1, x_2 and $f(x_1)$ and $f(x_2)$

If $f(x_2) > f(x_1)$ then $L_{k+1} = L_k$

$R_{k+1} = x_2$

If x_1 is minimum preserve x_1 and take left and right value ($L = a$, $R = x_2$)

If $f(x_2) < f(x_1)$ then $L_{k+1} = x_1$

$R_{k+1} = R_k$

If x_2 is minimum preserve x_2 and take left and right value as $L = x_1$, $R = b$.

Problems

PAGE NO :
DATE :

1. minimize $f(x) = x^2$, over $[-5, 15]$ using Fibonacci search by taking $n = 7$.

Solution

$f(x) = x^2$

$a = -5$, $b = 15$ $\quad L_0 = b - a$

$L_0 = 15 - (-5)$

$L = a = -5$. $R = b = 15$ $\quad L_0 = 20$

Fibonacci series

$$L_2^* = \frac{F_{n-2}}{F_n} L_0 = \frac{F(5)}{F7}(10) = \frac{8}{21}(20)$$

$$\Rightarrow L_2^* = 7.619$$

$x_1 = a + L_2^*$

$x_2 = b - L_2^*$

L	R	x_1	x_2	$f(x_1)$	$f(x_2)$	LoR
-5	15	2.619	7.3810	6.8592	54.47	L
		(L+R-x2)	(L+R-x1)			
-5	7.3810	-0.2380	2.619	0.0566	6.8592	L
-5	2.619	-2.1429	-0.2381	4.5920	0.0567	R
-2.1429	2.619	-0.2381	0.7142	0.0567	0.5101	L
-2.1429	0.7142	-1.1906	-0.2381	1.4175	0.0567	R
-1.1906	0.7142	-0.2381	-0.2383	0.0567	0.0568	R
-0.2381	0.7142	-0.2383	0.7144	0.0568	0.5104	

L = -0.2381, R = 0.7142

Preserve the Minimizer out of x_1 and x_2. L if $f(x)$ is minimum
R if $f(x)$ is minimum.

PAGE NO :
DATE :

$$x^* = \frac{-0.2381 + 0.7142}{2}$$

$$x^* = 0.2381$$

$$f(x^*) = 0.0567$$

2

solution

Retain L, find x_2

x_2 is R

PAGE NO :
DATE :

2. minimize $f(x) = x^2 + \frac{54}{x}$, [0 5] using Fibonacci search by taking $n = 3$

solution $f(x) = x^2 + \frac{54}{x}$

$a = 0$, $b = 5$, $n = 3$

$L = a = 0$

$R = b = 5$

$L_0 = b - a$

$L_0 = 5 - 0$

$L_0 = 5$

$L_2^* = \frac{F_{n-2}}{F_n}(L_0) = \frac{F_1}{F_3}(5) = \frac{1}{3}(5)$

$L_2^* = \cancel{2.5}\ 1.6667$

$x_1 = a + L_2^*$

$x_2 = b - L_2^*$

L	R	x_1	x_2	$f(x_1)$	$f(x_2)$	L (or) R
0	5	1.667	3.333	35.1724	27.3105	R
1.667	5	3.333	3.334	27.3105	27.3123	L
1.667	3.334	1.668	3.333	35.1563	27.3105	

$L = 1.667$, $b = 3.334$

$X^* = 2.5605$ $f(X^*) = 27.8481$

DATE:

4. Minimize $f(x) = x(x-1.5)$ in $[0\ 1]$ within the interval of uncertainity 0.25 of the initial interval of uncertainity

Solution

$$L_n \leq 0.25 L_0$$

$$\frac{L_n}{L_0} \leq 0.25$$

$$\frac{1}{F_n} \leq 0.25$$

$$\frac{1}{F_n} \geq 0.25$$

$$F_n \geq 4$$

1 1 2 3 5

$n = 4$

$$L = a = 0$$

$$R = b = 1$$

$$L_0 = b - a$$

$$L_0 = 1 - 0$$

$$L_0 = 1$$

$$L_2^* = \frac{F_{n-2}}{F_n}(L_0)$$

$$= \frac{F_{4-2}}{F_4}(1)$$

$$= \frac{F_2}{F_n}(1)$$

L	R
0	1
0.4	1
0.6	1
0.6	0

PAGE NO :
DATE :

$= \frac{2}{5}(1)$

$L_2^* = 0.4$

$x_1 = a + L_2^*$

$x_2 = b - L_2^*$

L	R	x_1	x_2	$f(x_1)$	$f(x_2)$	L/R
0	1	0.4	0.6	-0.44	-0.54	R
0.4	1	0.6	0.8	-0.54	-0.56	R
0.6	1	0.8	0.8	-0.56	-0.56	L
0.6	0.8					

$x^* = \frac{0.6 + 0.8}{2}$

$x^* = 0.70$

$f(x^*) = 0.70(0.7 - 1.5)$

$= -0.56$

DATE :

5. Minimize $f(x) = \frac{1-x}{2}$, $x < 1/2$

x^2 , $x \geq 1/2$

over $[-1, 1]$ perform six iterations.

Solution $L = a = -1$, $n = 6$

$R = b = 1$

$L_0 = b - a$

$= 1 - (-1)$

$= 1 + 1$

$L_0 = 2$

$$L_2^* = \frac{F_{n-2}}{F_n} (L_0)$$

$$= \frac{F_4}{F_6} (L_0)$$

$$= \frac{5}{13} (2)$$

$L_2^* = 0.7692$

$x_1 = a + L_2^*$

$x_2 = b - L_2^*$

PAGE NO :
DATE :

L	R	x1	x2	f(x1) (1st func)	f(x2) (1st func)	L/R
-1	1	-0.2308	0.2308	0.6154	0.3846	R
				(1st func)	(2nd)	
-0.2308	1	0.2308	0.5384	0.3846	0.2898	R
				(2nd)	(2nd)	
0.2308	1	0.5384	0.6924	0.2898	0.4794	L
				(1st)	(2nd)	
0.2308	0.6924	0.3848	0.5384	0.3076	0.2898	R
				(2nd)	(2nd)	
0.3848	0.6924	0.5384	0.5388	0.2898	0.2903	L
0.3848	0.5388					

$$x^* = \frac{0.3848 + 0.5388}{2}$$

$$x^* = 0.4618$$

1st Func

$$f(x^*) = 0.2691$$

DATE :

⇒ Local and Global optima :

→ Optimization refers to finding the set of inputs to an objective function that results in the maximum or minimum output

* General Form of Optimization

Minimize x (or Maximum x) $f_0(x)$

objective function.

Subject to the constraints $f_i(x) \leq (or =)$ (OR $\geq$) 0 for $i = 1, 2 \ldots, m$.

* Local Optima

A local optimum refers to a point within the domain of a function where the function attains lowest or highest value in its local neighbourhood. mathematically a point X^* is said to be a local minimum or maximum if there exists a neighbourhood around X^* such that $f(X^*) \leq f(x)$ → Local minimum

$f(X^*) \geq f(X)$ → maximum

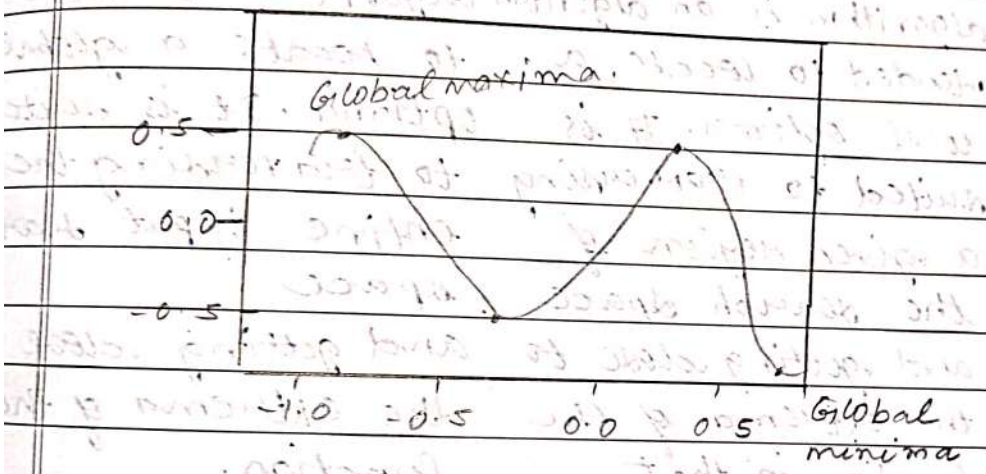
For all ($\forall$) X within that neighbourhood

In other words a local optimum is a point where the function is lower or higher than its neighbouring points.

PAGE NO :
DATE :

i. Global optima

The Global optimum of a function refers to the point within its domain where the function attains the lowest or highest value over the entire function. In other words it is the overall best solution for the optimization problem.



Difference between Local optima and Global optima

A local optimization algorithm should be used when you know that you are in the region of global optima or your objective function contains single optima	A Global optimization algorithm should be used when you know very little about structure of objective function, response surface or when you know that the function contains local optima

2. Pmax or min	It is the extrema of the objective function for a given region of the input space.	A Global optima is the extrema of objective function for the entire input search space
3.	A local optimization algorithm also called a local search algorithm is an algorithm intended to locate a local optima. It is suited to traversing a given region of the search space and getting close to the extrema of the function in that region.	A global optimization algorithm also called a global search algorithm is intended to locate a global optima. It is suited to traversing the entire input search space and getting close to the extrema of the function.
4.	Local optimization methods are widely used in applications where there is a value in finding a good point if not the very best	Global optimization is used for problems with a small number of variables where computing time is not critical and the value of finding the true global solution is very high.

PAGE NO :
DATE :

5. 3 Examples of local search algorithm using the definitions include

a) Nelder Mead Algorithm

b) BFGS algorithm

c) Hill climbing algorithm.

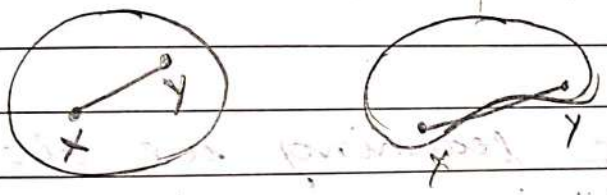
3 Examples of global search algorithm using the definitions include

a) Genetic algorithm

b) simulated - annealing algorithm

c) Particle - swarm optimization.

=> Convex sets



CONVEX SET NON-CONVEX SET

CONVEX SET

convexcity is a measure of the curvature and due to its curvature we can define two different concepts in convexcity which are convex sets and convex functions.

A convex set is a collection of points in which the line XY connecting any two points X, Y in the set lies completely within the set. Ex: Solid Cube

PAGE NO:
DATE:

A set is a convex set if we can draw a line segment between any two points in the set and all the points on that line segment are within that set.

Mathematically If $x, y \in C$ is convex

$\rightarrow \theta x + (1-\theta) y \in C \; \forall \; \theta \in [0,1]$

Ex: $\{ x / Ax = B \} = C$

$\{ x / Ax \leq B \} = C$

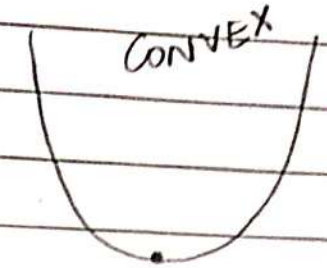
Question: Why convexity is important in machine learning.

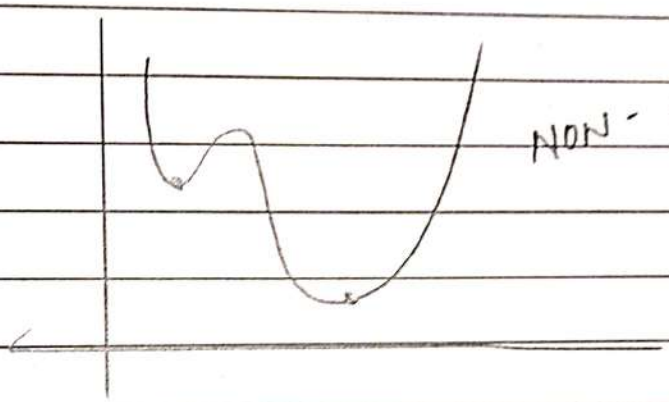
Answer: In machine learning we always want to minimize our objective function which is a loss function. convex functions are important in optimization because local minima of a convex function is also a global minima therefore once we find a local minimum our minimization problem is solved and we have found minimum loss.

CONVEX FUNCTION

A function is a convex function if its domain is a convex set and it is always below its secant segment.

PAGE NO :
DATE :





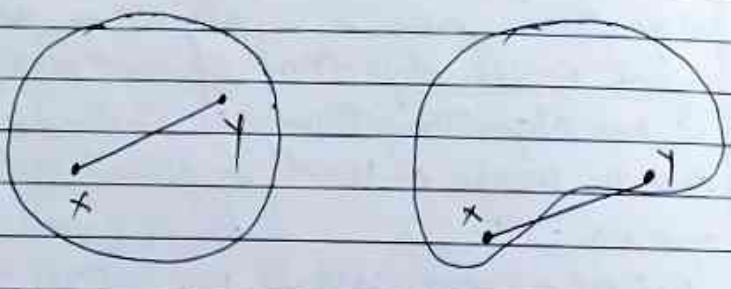
→ Three examples of local search algorithms using the definitions include;

(i) Nelder Mead algorithm
(ii) BFGS algorithm
(iii) Hill climbing algorithm.

Three examples of global search algorithms using the definitions include;

- Genetic algorithm
- Simulated Annealing algorithm
- Particle Swarm optimization

* Convex Sets &-



Convex Set Non-Convex Set

'Convexity' is a measure of the curvature and due to its curvature we can define two different concepts in convexity which are
→ Convex sets
→ Convex functions.

A Convex Set is a collection of points in which the line XY connecting any two points X, Y in this set lies completely within the set.

PAGE NO
DATE

E.g: a 3D Solid cube

A set is a convex set if we can draw a line segment betn any two points in the set and all the points on that line segment are within that set.

Mathematically, if $x, y \in C$ is convex
$\Rightarrow \theta x + (1-\theta) y \in C \quad \forall \theta \in [0,1]$

Eg: $\{ x / Ax = B \} = C$

$\{ x / Ax \leq B \} = C$

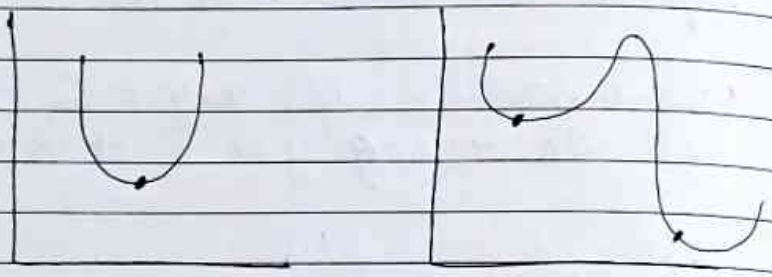
* Why Convexity is important in Machine Learning?

In ML, we always want to minimize our objective function (which is a loss function.)

Convex func. are important in optimization because a local minima of a convex func. is also a Global minima.

Therefore, once we find a local minima, our minimization problem is solved and we've found minimum loss.

Convex function is function where its domain is a convex set and is always below its secant-segment.



Convex func Non-Convex func.

Three-Point Interval Search :

24/06/24

This method is used for un-constraint optimization. In this method we divide the interval in 4 equal parts. We select the central point as a functional value which contains maxima or minima.

Then, we select its interval around its central functional value.
We repeat this process until we reach such a value that shows less than epsilon (ε) value.

$$|f(x_c) - f(x_{cnm})| < \varepsilon$$

The above step is called stopping criteria.

1. By using Three point interval search [TPIS] method to find maximum of $f(x) = x(5\pi - x)$ on $[0, 20]$ with $\varepsilon = 0.1$.

Sol^n: Given function : $f(x) = x(5\pi - x)$

Interval : $[0, 20]$

$A = 0$

$B = 20$

$n = 4$.

$= \frac{B-A}{n} = \frac{20-0}{4} = 5$.

a	x_0	x_1	x_2	b
0	05	10	15	20

$= f(a) = f(0) = 0(5\pi - 0) = 0$ //

$= f(x_0) = f(5) = 5(5\pi - 5) = 25\pi - 25$

$= 25(\pi - 1)$

$= 25(2.1416)$

$= 53.5398$ //

$= f(x_1) = f(10) = 10(5\pi - 10) = 57.0796$ //

$= f(x_2) = f(15) = 15(5\pi - 15) = 10.6194$ //

$= f(b) = f(20) = 20(5\pi - 20) = -85.8407$ //

The function has maximum value at $f(x_1) = 57.0796$

Central value $\Rightarrow f(x_1) = 57.07796$.
Sub-interval $\Rightarrow [5, 15]$.

$\Rightarrow \left| f(x_c) - f(x_{calm}) \right|$.

$|57.0796 - 53.5398|$

$= 3.5371 \not< \varepsilon \quad (\because \varepsilon = 0.1)$

iteration 1

Now interval $[a, b] = [5, 15]$
$n = 4$.

$\frac{b-a}{n} = \frac{15-5}{4} = \frac{10}{4} = 2.5$.

x_0	x_3	x_1	x_4	x_2
5	7.5	10	12.5	15

$f(x_3) = f(7.5) = 61.5597$

$f(x_4) = f(12.5) = 40.0995$

$f(x_0) = f(5) = 53.5398$
$f(x_1) = f(10) = 57.0796$
$f(x_2) = f(15) = 10.6194$.

Central value $\Rightarrow f(x_3) = 61.5597$

$x_3 = 7.5$

Sub-interval $= [x_0, x_1] = [5, 10]$

$|f(x_c) - f(x_{new})|$

$|61.5597 - 57.0796|$

$= 4.4801 \not< \varepsilon$

iteration 2

Now interval $[x_0, x_1] = [5, 10]$

$n = 4$

$\frac{b-a}{n} = 1.25.$

x_0	x_5	x_3	x_6	x_1
5	6.25	7.5	8.75	10

$f(x_5) = 59.1123$

$f(x_6) = 60.8822$

~~$f(x_{50}) =$~~

Here, $f(x_3) = f(7.5) = 61.5597$ is maximum value.

Central value $= x_3 = 7.5$

Sub-interval $= [x_5, x_6] = [6.25, 8.75]$

$|f(x_c) - f(x_{cNM})|$

$|61.5597 - 60.8822|$

$= 0.6775 \not< \varepsilon$

iteration 3

Interval is $[x_5, x_8] = (6.25 \to 8.75)$

$n = 4$

$\frac{b-a}{n} = 0.6250.$

x_5	x_7	x_3	x_8	x_6
6.25	6.875	7.5	8.125	8.75

$f(x_7) = 60.7266$

$f(x_8) = 61.6116.$

Central maximum value

$= f(x_8) = 61.6116.$

Sub-interval $= [x_3, x_6] = [7.5, 8.$

$|f(x_c) - f(x_{cNM})|$

$= |61.6116 - 61.5597|$

$= 0.0519 < \varepsilon$

∴ Maximum value of the function $f(x)$ is 61.6116 at $x_8 = 8.125$.

2. By using TPIS method to find max. of $f(x) = x + \frac{9}{x}$ on $[0,2]$ with $\varepsilon = 0.001$.

3. By using TPIS method to find max. of $f(x) = x \sin 4x$ on $[0,3]$ with $\varepsilon = 0.002$.

1/07/2020

Lagrangian Multipliers Method.

* **General problem:**

Maximize or minimize $f(x)$;

$$x = (x_1, x_2, x_3 \cdots x_n)^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Subject to $g_j(x) = 0$, $j = 1, 2 \cdots m$.

Then the Lagrangian function is given by

$$L(x, \lambda_j) = f(x) + \sum_{j=1}^{m} \lambda_j g_j(x)$$

— OR —

$$L(x_1, x_2 \cdots x_n, \lambda_1, \lambda_2 \cdots \lambda_m) = f(x_1, x_2 \cdots x_n) + \lambda_1 g_1(x) + \lambda_2 g_2(x) + \cdots + \lambda_m g_m(x).$$

— OR —

$$L(x_i, \lambda_j) = f(x_i) + \sum_{j=1}^{m} \lambda_j g_j(x_i), \quad i = 1, 2 \cdots n, \quad j = 1, 2, \cdots m.$$

where $\lambda_j \geq 0$ are called Lagrangian Multipliers.

PAGE NO:
DATE:

* Necessary Conditions :

For a function $f(x)$ to be max. or min.

$$\frac{dL}{dx_i} = 0 \Rightarrow \frac{df}{dx_i} + \sum_{j=1}^{m} \lambda_j \frac{dg_j(x)}{dx_i} = 0 \quad \text{--- (1)}$$

and

$$\frac{dL}{d\lambda_j} = g_j(x) = 0 \quad \text{--- (2)}$$

* Suffecient Conditions :

Let x_0 be a stationary point which can be obtained from solving equations (1) and (2)

Construct the bordered Hessian matrix

$$H_B = \begin{bmatrix} 0 & \nabla g \\ (\nabla g)^T & L_H \end{bmatrix}_{(m+n)\times(m+n)} \quad \text{(order)}$$

where,

$$\nabla g = \left(\frac{dg_j}{dx_i}\right)_{m\times n} \quad \& \quad L_H = \left[\frac{d^2L(x,\lambda)}{dx_i\,dx_j}\right]_{n\times n}$$

$n \longrightarrow$ no. of variables
$m \longrightarrow$ no. of constraints.

NOTE : The stationary point x_0 is a min. point if starting with the principle minor determinant of order $(2m+1)$. The last $(n-m)$ principle minors of D th have the sign of $(-1)^m$.

The stationary point x_0 is the max. point if starting with the principle minor determinant of order $(2m+1)$. The last $(n-m)$ principle minors of P have the alternating sign pattern starting with $(-1)^{m+1}$.

Problems :-

Optimize

~~Minimize~~ $f(x) = 3x_1^2 + 4x_2^2 + 5x_3^2$ subject to $x_1 + x_2 + x_3 = 10$.

Representing the given function $f(x)$ in the std. Lagrangian form.

$$L(x, \lambda) = 3x_1^2 + 4x_2^2 + 5x_3^2 + \lambda(x_1 + x_2 + x_3 - 10)$$

$n = 3$ (x_1, x_2, x_3)

$m = 1$ (subject to $x_1 + x_2 + x_3 = 10$)

Necessary Conditions.

(Equating to zero)

$$\frac{dL}{dx_1} = 6x_1 + \lambda = 0 \Rightarrow \boxed{x_1 = -\lambda/6}$$

$$\frac{dL}{dx_2} = 8x_2 + \lambda \Rightarrow \boxed{x_2 = -\lambda/8}$$

$$\frac{dL}{dx_3} = 10x_3 + \lambda \Rightarrow \boxed{x_3 = -\lambda/10}$$

$$\frac{dL}{d\lambda} = x_1 + x_2 + x_3 - 10 \Rightarrow \boxed{x_1 + x_2 + x_3 = 10}$$

6, 8, 10
5, 4, 3

PAGE NO.
DATE

Substituting values of x_1 x_2 x_3.

$$\frac{(-\lambda)}{6}^{\times 20} + \frac{(-\lambda)}{8}^{15} + \frac{(-\lambda)}{10}^{12} = 10$$

$$\frac{-20\lambda - 15\lambda - 12\lambda}{120}$$

$$\frac{-47\lambda}{120} = 10.$$

$$\Rightarrow \quad \lambda = \frac{-1200}{47}$$

$$x_1 = \frac{-\lambda}{6} = \frac{1200}{47 \times 6} = \frac{200}{47}$$

$$x_2 = \frac{-\lambda}{8} = \frac{1200}{47 \times 8} = \frac{150}{47}$$

$$x_3 = \frac{-\lambda}{10} = \frac{1200}{47 \times 10} = \frac{120}{47}.$$

$$x_0 = x^* = (x_1, x_2, x_3)$$

$$x_0 = x^* = \left(\frac{200}{47}, \frac{150}{47}, \frac{120}{47}\right).$$

Suffecient conditions:-

$$Dg = \begin{bmatrix} \frac{dg}{dx_1} & \frac{dg}{dx_2} & \frac{dg}{dx_3} \end{bmatrix}$$

$$g(x) \quad x_1 + x_2 + x_3 - 10.$$

for question. Optimize → sufficient conditions to be done
minimize/maximize → not req.

PAGE NO
DATE

$$\nabla g = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$\nabla g_{\text{at } x_0} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$(\nabla g)^T = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \nabla g_{\text{at } x_0} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$L_H = \begin{bmatrix} \frac{d^2L}{dx_1^2} & \frac{d^2L}{dx_1 dx_2} & \frac{d^2L}{dx_1 dx_3} \\ \frac{d^2L}{dx_2 dx_1} & \frac{d^2L}{dx_2^2} & \frac{d^2L}{dx_2 dx_3} \\ \frac{d^2L}{dx_3 dx_1} & \frac{d^2L}{dx_3 dx_2} & \frac{d^2L}{dx_3^2} \end{bmatrix}$$

$$L_H = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 10 \end{bmatrix} \qquad L_{H \text{ at } x_0} = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

$$H_B = \begin{bmatrix} 0 & \nabla g \\ (\nabla g)^T & L_H \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 6 & 0 & 0 \\ 1 & 0 & 8 & 0 \\ 1 & 0 & 0 & 10 \end{bmatrix}$$

$$\Delta_3 = \begin{vmatrix} 0 & 1 & 1 \\ 1 & 6 & 0 \\ 1 & 0 & 8 \end{vmatrix} = 0(\,) - (1)(8-0) + 1(0-6)$$

$$= -14$$

Δ_3 Δ_4 → Last two (n-m) principle minors.

$$\Delta_4 = \begin{vmatrix} 6 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 10 \end{vmatrix} - 1 \begin{vmatrix} 1 & 0 & 0 \\ 1 & 8 & 0 \\ 1 & 0 & 10 \end{vmatrix} +$$

$$1 \begin{vmatrix} 1 & 6 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 10 \end{vmatrix} - 1 \begin{vmatrix} 1 & 6 & 0 \\ 1 & 0 & 8 \\ 1 & 0 & 0 \end{vmatrix}$$

$= 0(\quad) - 1[1(80-80) - 0(\quad) + 0(\quad)]$

$+ 1[1(0-0) - 6(10-10) + 0(\quad)] - 1$

$[1(0-0) - 6(8-8) + 0(\quad)]$

$= -80 + 60 - 48$

$= -188.$

$f(x_2) = 3x_1^2 + 4x_2^2 + 5x_3^2$

$= 3\left(\frac{200}{47}\right)^2 + 4\left(\frac{150}{47}\right)^2 + 5\left(\frac{120}{47}\right)^2$

$= 54.3232 + 40.7424 + 32.5939$

$= 127.6595.$

PAGE NO:
DATE:

2. Minimize $f(x) = x_1^2 + x_2^2$ subject to $a_1x_1 + a_2x_2 = b$

$n = 2 \quad m$

Std. Lagrangian form.

$$L(x, \lambda) = x_1^2 + x_2^2 + \lambda(a_1x_1 + a_2x_2 - b)$$

Necessary conditions.

$$\frac{dL}{dx_1} = 2x_1 + \lambda a_1 \qquad x_1 = -\lambda a_1/2$$

$$\frac{dL}{dx_2} = 2x_2 + \lambda a_2 \qquad x_2 = -\lambda a_2/2$$

$$\frac{dL}{d\lambda} = a_1x_1 + a_2x_2 - b \qquad a_1x_1 + a_2x_2 = b.$$

$$= a_1(-\lambda a_1/2) + a_2(-\lambda a_2/2) = b$$

$$= \frac{-\lambda a_1^2}{2} - \frac{\lambda a_2^2}{2} = b.$$

$$-\lambda\left(\frac{a_1^2 + a_2^2}{2}\right) = b$$

$$\boxed{-\lambda = \frac{2b}{a_1^2 + a_2^2}}$$

$$x_1 = \frac{-\lambda}{2} = \frac{2b\,(a_1)}{(a_1^2 + a_2^2)\,2} = \frac{ba_1}{a_1^2 + a_2^2}$$

$$\lambda_2 = \frac{2b(a_2)}{(a_1^2+a_2^2)2} = \frac{ba_2}{a_1^2+a_2^2}$$

Min. point or Stationary point $x_0 = x^*$

$$= \left(\frac{a_1 b}{a_1^2+a_2^2}, \frac{a_2 b}{a_1^2+a_2^2}\right)$$

$$f(x^*) = f(x_0) = \frac{b^2}{a_1^2+a_2^2} \rightarrow \frac{a_1^2b^2}{(a_1^2+a_2^2)^2} + \frac{a_2^2b^2}{(a_1^2+a_2^2)^2} \rightarrow \frac{(a_1^2+a_2^2)b^2}{(a_1^2+a_2^2)^2}$$

3. Minimize $Z = x_1^2 + x_2^2 + x_3^2$ subject to conditions

$$x_1 + x_2 + 3x_3 = 2,$$
$$5x_1 + 2x_2 + x_3 = 5.$$

$$L(x_1 \lambda) = x_1^2 + x_2^2 + x_3^2 + \lambda_1(x_1 + x_2 + 3x_3 - 2) + \lambda_2(5x_1 + 2x_2 + x_3 - 5)$$

$$\frac{dL}{dx_1} = 2x_1 + \lambda_1 + 5\lambda_2 = 0 \Rightarrow 2x_1 = -5\lambda_2 - \lambda_1$$

$$x_1 = \frac{-\lambda_1 - 5\lambda_2}{2}$$

$$x_1 = -\lambda_1/2 - 5\lambda_2/2$$

$$\frac{dL}{dx_2} = 2x_2 + \lambda_1 + 2\lambda_2$$

$$x_2 = \frac{-\lambda_1 - 2\lambda_2}{2}$$

$$\frac{dL}{dx_3} = 2x_3 + 3\lambda_1 + \lambda_2$$

$$x_3 = \frac{-3\lambda_1 - \lambda_2}{2}$$

$$\frac{dL}{d\lambda_1} = x_1 + x_2 + 3x_3 - 2 = 0$$

$$= \left(\frac{-\lambda_1 - 5\lambda_2}{2}\right) + \left(\frac{-\lambda_1 - 2\lambda_2}{2}\right) + 3\left(\frac{-3\lambda_1 - \lambda_2}{2}\right) - 2 = 0$$

$$= -\lambda_1 - 5\lambda_2 + -\lambda_1 - 2\lambda_2 (-3\lambda_1 - \lambda_2)^3 - 4 = 0$$

$$= -11\lambda_1 - 10\lambda_2 - 4 = 0 \quad \text{— (1)}$$

$$\frac{dL}{d\lambda_2} = 5\left(\frac{-\lambda_1 - 5\lambda_2}{2}\right) + 2\left(\frac{-\lambda_1 - 2\lambda_2}{2}\right) + \left(\frac{-3\lambda_1 - \lambda_2}{2}\right) - 5 = 0$$

$$= -10\lambda_1 - 30\lambda_2 - 10 = 0. \quad \text{— (2)}$$

Solving;

$$\lambda_1 = -2/23 \qquad \lambda_2 = -7/23$$

$$x_1 = 37/46$$

$$x_2 = 16/46$$

$$x_3 = 13/46$$

PAGE NO :
DATE :

$$x^* = \left(\frac{27}{46}, \frac{16}{46}, \frac{13}{46}\right)$$

$$f(x^*) = \frac{39}{46}.$$

4. Maximize $Z = 2x_1^2 + x_2^2 + 3x_3^2 + 10x_1 + 8x_2 + 6x_3 - 100.$

subject to $x_1 + x_2 + x_3 - 20 = 0.$

Explain Stochastic Gradient Descent method.

It is an Iterative method for optimising an objective function by updating the model parameters, to minimise a loss function.
It a Stochastic approximation of gradient descent optimization which replaces the actual gradient with an estimate.
Stochastic gradient is computationally efficient and can converge faster that batch gradient descent. But it can be noisy and may not converge to global minimum.
Stochastic Gradient randomly divides the set of observations into mini batches. For each mini batch the gradient is computed and the vector is moved. once all minibatches are used we say that the iteration is finished and start the next one

Algorithm:

1. Start with an initial set of parameters
2. Shuffle the training data set to introduce randomness
3. Randomly select training from the Sheffled dataset
4. Compute the gradient of the cost function wrt the model parameters using the current training data set
5. Update the model parameters by taking a step in the direction of the negative gradient scaled by the learning rate.
6. Repeat the process for multiple iterations until the model converges or reaches a predefined stoping criterian

Differences B/w Stochastic gradient descent and Minibatch gradient descent method.

Stochastic gradient descent	Minibatch gradient descent
1. Sgd uses a single data point or small batch	1. Minibatch uses a small subset of data.
2. Updates parameters after each observation using a small batch	2. Updates parameters after each subset of data
3. Faster than gradient descent method especially for large data set	3. More efficient than sgd and less computationally intensive than batch gradient descent. method.
4. Sgd forms zigzag convergence graph	4. It forms a smoother convergence graph than sgd. but not as much as batch gradient descent
5. Suitable for large data set	5. Often used in practise and considered a good compromise b/w Sgd and batch gradient descent.

Advantages and disadvantages of Stochastic gradient descent method :-

Advantages :-

*Speed : It is faster than other variance of gradient descent such has batch gradient descent. and mini batch gradient descent since it uses only one example to update the parameters

*Memory efficiency : Since it updates the parameters for each training example one at a time. It is memory efficient and can

handle large data set that cannot fit into memory.

*Avoidance of Local Minima :- Due to the noisy updates in Sgd it has the ability to escape from local minima and converges to a global minima.

Disadvantages :

* Noisy updates : The updates in Sgd are noisy and have a high variance which can be the optimization progress. less stable and lead to oscillations around the minimum.

* Slow convergence :- Sgd ~~makes~~ may require more iterations to converge to the minimum since it updates the parameters for each train example one at a time

* Sensitivity to learning rate :- The choice of learning rate can be critical in sgd since using high learning rate can cause the algorithm to overshoot the minimum while a low learning rate can make the algorithm converge slowly.

* less Accurate :- Due to the noisy update Sgd may not converge to the exact global minimum and can result in suboptimal solution - This can be technique such has learning rate scheduling and moment based updates.

DOMS | Page No.
Date / /

Newton-Raphson Method:

1. Choose initial value x_0 and a small number ε if given, if not, consider $\varepsilon = 10^{-4}$, then compute $f'(x_0)$
2. Compute $f''(x_0)$
3. Calculate $x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$
4. Compute $f'(x_{i+1})$.
5. If $|f'(x_{i+1})| < \varepsilon$, then terminate the process else set $i = i+1$, go to step2.

PRO

1. Use NR method to approximate $f(x) = 2\sin x - \frac{x^2}{10}$ with initial guess of 2.5 & $\varepsilon = 10^{-4}$.

Given:

$f(x) = 2\sin x - \frac{x^2}{10}$, $x_0 = 2.5$, $\varepsilon = 10^{-4}$

$f'(x) = 2\cos x - \frac{2x}{10} = 2\cos x - \frac{x}{5}$

$f''(x) = -2\sin x - \frac{1}{5}$

*calci must be in R

i. 1st iteration: $i = 0$

$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} \Rightarrow 2.5 - \frac{-2.10229}{-1.39694} = 0.99502$

~~$\frac{1.4981}{0.28724}$~~

$x_1 = 0.995$

$f'(x_1) = |0.88997| \not< 10^{-4}$

ii. 2nd iteration: $i = 1$

$x_2 = x_1 - \frac{f'(x_1)}{f''(x_1)} \Rightarrow 0.88997 - \frac{0.99502 - 0.88997}{-1.87754}$

$x_2 = 1.46903$

$f'(x_2) = |-0.09062| \not< 10^{-4}$

iii. 3rd iteration: $i = 2$

$x_3 = x_2 - \frac{f'(x_2)}{f''(x_2)} \Rightarrow 1.46903 - \frac{(-0.09062)}{-2.18965}$

$x_3 = 1.42764$

$f'(x_3) = |-0.00019| \not< 10^{-4}$

iv. 4th iteration: $i = 3$

$x_4 = x_3 - \frac{f'(x_3)}{f''(x_3)} \Rightarrow 1.42764 - \frac{(-0.00019)}{-2.17954}$

$x_4 = 1.42755$

$f'(x_4) = |0.000003877| < 10^{-4}$ ✓

$\therefore f'(x_4) = 0.000003877$ for $x_4 = 1.42755$.

$\therefore f(x_4) = 1.77573$ //

2. Use NR method to approximate $f(x) = x^2 + \frac{54}{x}$, $x_0 = 2.8$, $\varepsilon_0 = 10^{-4}$

Given:

$f(x) = x^2 + \frac{54}{x}$; $x_0 = 2.8$, $\varepsilon_0 = 10^{-4}$

$f'(x) = 2x - \frac{54}{x^2}$; $f''(x) = 2 + \frac{54}{x^3}$

DOMS | Page No.
Date / /

$f'(x) = 2x - \frac{54}{x^2}$; $f'(x) = 2 + \frac{108}{x^3}$ $\quad 2 + \frac{108}{x^3}$

i. iteration 1 : $i = 0$

$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} \Rightarrow 2.8 - \frac{(-40.56)}{6.91983} = 11.8944$

$x_1 = 11.8944$

$\Rightarrow 2.8 - \frac{(-1.28776)}{6.91983} = 2.9861$

$x_1 = 2.9861$

$f'(x_1) = |-0.08379| \not< 10^{-4}$

ii. iteration 2 : $i = 1$

$x_2 = x_1 - \frac{f'(x_1)}{f''(x_1)} \Rightarrow 2.9861 - \frac{(-0.08379)}{6.05612}$

$x_2 = 2.999936$

$f'(x) = |-0.000384| \not< 10^{-4}$

iii. iteration 3 : $i = 2$

$x_3 = x_2 - \frac{f'(x_2)}{f''(x_2)} \Rightarrow 2.999936 - \frac{(-0.000384)}{6.000256}$

$x_3 = 2.9999999997$

$f'(x_3) = |-0.0000000016| < 10^{-4}$

$\therefore f'(x_3) = |-0.0000000016| < 10^{-4}$ for $x_3 = 2.9999999997$

$\therefore f(x) = 27$ //

DOMS | Page No.
Date / /

3. Find the extrema of the function using NR method

Given : $f(x) = 5\sin(2x) - 2x^2 - 4x$,

$\varepsilon = 10^{-4}$, $x_0 =$ ② (is not given, take 0).

$f'(x) = 5\cos(2x) - 4x - 4$

$= 10\cos(2x) - 4x - 4$

$f''(x) = -20\sin(2x) - 4$

Find

Given: $f(x) = 5\sin(2x) - 2x^2 - 4x$

$x_0 = -2, \quad \varepsilon = 10^{-4}$

$f'(x_0) = 10\cos(2x) - 4x - 4$

$f''(x) = -20\sin(2x) - 4$

i. iteration 1 : i = 0

$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} \Rightarrow -2 - \frac{(-2.5364)}{-19.13605}$$

$x_1 = -2.1325$

$f'(x_1) = |0.20387| \not< 10^{-4}$

ii. iteration 2 : i = 1

$$x_2 = x_1 - \frac{f'(x_1)}{f''(x_1)} \Rightarrow -2.1325 - \frac{(0.20387)}{-22.03159}$$

$x_2 = -2.1232$

$f'(x_2) = |0.00192| \not< 10^{-4}$

iii. iteration 3 : i = 2

$$x_3 = x_2 - \frac{f'(x_2)}{f'(x_2)} \Rightarrow -2.1233 - \frac{(0.00192)}{-21.86935}$$

$x_3 = -2.1232$

$f'(x_3) = |-0.0002655| \not< 10^{-4}$

DOMS | Page No.
Date / /

iv. iteration 4 : i = 3

$$x_4 = x_3 - \frac{f'(x_3)}{f''(x_3)} \Rightarrow -2.1232 - \frac{(-0.0002655)}{-21.86755}$$

$$x_4 = -2.12321$$

$$f'(x_4) = |-0.0000000| < 10^{-4} \text{ //}$$

Optimization Techniques (BCS405C)

Module-5: Advanced Optimization

RMSprop:

RMSProp (Root Mean Squared Propagation) is an adaptive learning rate optimization algorithm. It is an extension of the popular Adaptive Gradient Algorithm and is designed to dramatically reduce the amount of computational effort used in training neural networks. This algorithm works by exponentially decaying the learning rate every time the squared gradient is less than a certain threshold. This helps reduce the learning rate more quickly when the gradients become small. In this way, RMSProp is able to smoothly adjust the learning rate for each of the parameters in the network, providing a better performance than regular Gradient Descent alone.

The RMSprop algorithm utilizes exponentially weighted moving averages of squared gradients to update the parameters. Here is the mathematical equation for RMSprop:

1. Initialize parameters:
   - Learning rate: α
   - Exponential decay rate for averaging: γ
   - Small constant for numerical stability: ε
   - Initial parameter values: θ
2. Initialize accumulated gradients (Exponentially weighted average):
   - Accumulated squared gradient for each parameter: $E_t = 0$
3. Repeat until convergence or maximum iterations:
   - Compute the gradient of the objective function with respect to the parameters: $g_t = \nabla_\theta J(\theta_t)$
   - Update the exponentially weighted average of the squared gradients: $E_t = \gamma E_{t-1} + (1-\gamma) g_t^2$
   - Update the parameters: $\theta_{t-1} = \theta_t - \alpha \frac{g_t}{\sqrt{E_{t+\epsilon}}}$

where,

- g_t is the gradient of the loss function with respect to the parameters at time *t*
- γ is a decay factor
- E_t is the exponentially weighted average of the squared gradients
- α is the learning rate
- ϵ is a small constant to prevent division by zero

This process is repeated for each parameter in the optimization problem, and it helps adjust the learning rate for each parameter based on the historical gradients. The exponential moving average allows the algorithm to give more importance to recent gradients and dampen the effect of older gradients, providing stability during optimization.

Stochastic Gradient Descent with Momentum:

SGD	SGD with Impulse
$\theta_j \leftarrow \theta_j - \epsilon \nabla_{\theta_j} L(\theta)$	$v_{t+1} \leftarrow \rho v_t + \nabla_\theta L(\theta)$
	$\theta_j \leftarrow \theta_j - \epsilon v_{t+1}$

GL. 2 Stochastic GD (left), SGD with momentum (right).

On the left side in GL. 2 is the formula for the weight updates according to the regular stochastic gradient descent (SGD for short). The equation on the right represents the rule for

the updates of the weights according to the SGD with momentum. **Momentum appears here as an additional term** $\rho.v$, which is added to the regular update rule.

Intuitively speaking, by adding this impulse term, we let our **gradient build up some sort of velocity** v during training. The velocity is the running sum of the gradients weighted by ρ.

The parameter ρ can be thought of as friction that "slows" the velocity down a bit. In general, velocity can be seen to increase with time. By using the momentum term, **saddle points and local minima become less dangerous** for the gradient. This is because the step size toward the global minimum now depends not only on the slope of the loss function at the current point, but **also on the velocity** that has built up over time.

If you would like a physical representation of stochastic gradient descent with momentum, please imagine a ball rolling down a hill, increasing in velocity with time. If this ball encounters an obstacle along the way, such as a hole or flat ground with no slope, its built-up velocity v would give the ball enough force to roll over this obstacle. In this case, the **flat ground represents a saddle point** and the **hole represents a local minima** of a loss function.

Adagrad:

Another optimization strategy is called as AdaGrad. The idea behind AdaGrad is that you keep a running sum of squared gradients during optimization. In this case, we don't have a momentum term, but an expression g_{t+1}, which is **the sum of squared gradients** up to the time $(t+1)$.

SGD with Impulse	Adagrad
$v_{t+1} \leftarrow \rho v_t + \nabla_\theta L(\theta)$	$g_0 = 0$
$\theta_j \leftarrow \theta_j - \epsilon v_{t+1}$	$g_{t+1} \leftarrow g_t + \nabla_\theta L(\theta)^2$
	$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta L}{1e^{-5}+\sqrt{g_{t+1}}}$

The updating rule of parameters at AdaGrad

When we optimize a weights θ_j, we divide the current gradient $\nabla_j L$ by the root of the term g_{t+1} . To understand the intuition behind AdaGrad, please imagine a loss function in a two-dimensional space. In this space, the gradient of the loss function **increases very weakly in one direction** and **very strongly in the other direction**. If we now sum up the gradients along the axis in which the gradients increase weakly, the squared sum of these gradients becomes even smaller.

If during the update step we divide the current gradient $\nabla_j L$ by a very small sum of the squared gradients g_{t+1} , the quotient becomes very high. For the other axis, along which the gradients increase sharply, exactly the opposite is true.

This means that we speed up the updating process **along the axis with weak gradients** by increasing these gradients along this axis. On the other hand, we slow down the updates of the weights **along the axis with large gradients**.

However, there is a problem with this optimization algorithm.

Imagine what would happen to the sum of squared gradients if the training takes too long. Over time, this term would **grow larger.** When the current gradient is divided by this large number, the update step for the weights becomes very small. It is as if we were using **a very low learning rate**, which becomes even lower the longer the training takes. In the worst case, we would get stuck at AdaGrad and the training would go on forever.

ADAM:

So far, we have used the momentum term to determine the velocity of the gradient and update the weight parameter in the direction of that velocity. In the case of AdaGrad and RMSProp, we used the sum of squared gradients to scale the current gradient so that we could update the weights in each space dimension with the same ratio.

These two methods seemed like pretty good ideas. Why don't we just **take the best of both worlds** and combine these ideas into a single algorithm? That's exactly the concept behind the final optimization algorithm I'd like to introduce. This algorithm is called ADAM.

The main part of this optimization algorithm consists of the following three equations. These equations may seem complicated at first glance, but if you look closely, you will see some similarities with the last three optimization algorithms.

The main part of this optimization algorithm consists of the following three equations. These equations may seem complicated at first glance, but if you look closely, you will see some similarities with the last three optimization algorithms.

$$m_0 = 0 \; and \;\; v_0 = 0$$

$$m_{t+1} \leftarrow \beta_1 m_t + (1-\beta_1)\nabla_\theta L(\theta) \text{ ---------- Impulse}$$

$$v_{t+1} \leftarrow \beta_2 v_t + (1-\beta_2)\nabla_\theta L(\theta)^2 \text{----------RMS prop}$$

$$\theta_j \leftarrow \theta_j - \frac{\epsilon}{\sqrt{v_{t+1}} + 1e^{-5}} m_{t+1} \text{---------- RMSprop+ Impulse}$$

Eq. 1 Parameter update rule for ADAM Optimizer

The first expression in Eq.1 looks a bit like SGD with momentum. In this case, the term m_t would be the velocity and the term β_1 would be the friction term. In the case of ADAM, we refer to m_t as the "first momentum", β_1, on the other hand, is just a hyperparameter. However, the difference with SGD with momentum is the factor $(1-\beta_1)$multiplied by the current gradient.

The second expression in Eq.1 **can be considered as RMSProp**, where we keep the running sum of squared gradients. Also in this case, there is the factor$(1-\beta_2)$, which is multiplied by the squared gradient.

The term v_t in Eq.1 is called the "second momentum" and is also just a hyperparameter. The final update equation (third term in the equation) can be viewed as **a combination of RMSProp and SGD with momentum**.

ADAM has incorporated the nice features of the previous two optimization algorithms. However, there is a small problem with ADAM, and that is what happens at the beginning of training….

At the very first time step $= 0$, the first and second pulse terms m_0 and v_0 are set to zero. After the first update of the second momentum v_1 , this term is still very close to zero. When

we update the weight parameters in the last expression in equation, we divide by a very small second momentum v_1 . This leads to a very large first update step.

This very large first update step is not the result of the geometry of the problem, but an artefact of the fact that we initialized the first and second momentum to zero. To address the problem of large update steps happening at the beginning of training, ADAM includes a correction clause:

$$m_{t+1} \leftarrow \beta_1 m_t + (1-\beta_1)\nabla_\theta L(\theta) \text{ ---------- Impulse}$$

$$v_{t+1} \leftarrow \beta_2 v_t + (1-\beta_2)\nabla_\theta L(\theta)^2 \text{----------RMS prop}$$

$$\hat{m}_{t+1} \leftarrow \frac{m_{t+1}}{1-\beta_1^t} \text{ and } \hat{v}_{t+1} \leftarrow \frac{v_{t+1}}{1-\beta_2^t} \text{ ----------Bias corrector}$$

$$\theta_j \leftarrow \theta_j - \frac{\epsilon}{\sqrt{v_{t+1}} + 1e^{-5}} m_{t+1} \text{---------- RMSprop+ Impulse}$$

Eq. 2 Bias Correction for ADAM Optimizer

As you can see from Eq.2 after the initial update of the first and second pulses, we **make an unbiased estimate of these pulses** by considering the current time step. With the so-called bias correction, we obtain the corrected first and second impulses with $\hat{m}_{t+1}$ and $\hat{v}_{t+1}$ respectively.

These corrections cause the values of the first and second impulse to be higher at the beginning of the training than without this correction. As a result, the first update step of the neural network weight parameters does not become too large. Thus, the training is not already messed up at the very beginning.

With the additional bias corrections, we obtain the complete form of the ADAM optimizer.

The best optimization algorithm for Machine Learning:

Now, we can discuss the question of which algorithm is the best to train a neural network.

In general, a normal gradient descent algorithm is **more than sufficient for simpler tasks**. If you are not satisfied with the accuracy of your model, you can try RMSprop or add a momentum term to your gradient descent algorithms.

However, **ADAM is the best neural network optimization algorithm** available today. This optimization algorithm is excellent for almost any deep learning problem you will ever encounter in practice. Especially if you set ADAM's hyperparameters to the following values:

- $\beta_1 = 0.9$
- $\beta_2 = 0.999$
- learning rate = 0.001 – 0.0001

ADAM and the above values of the hyperparameters are a very good starting point for any problem and virtually any type of neural network architecture.

For this reason, **ADAM is the default optimization algorithm** for any Deep Learning problem. Only in very few cases we can switch to other optimization algorithms like RMSprop, Adagrad etc..

It is recommended that we always start with ADAM, regardless of the neural network architecture of the problem domain.

What is saddle point problem in Machine Learning?

The term **"saddle point"** in the context of machine learning refers to a specific point in the optimization landscape of a cost function where the gradient is zero, but the point is neither a minimum nor a maximum. Instead, it's a point where the surface of the cost function resembles a saddle, with some dimensions curving upward and others downward.

Key Characteristics of a Saddle Point:

1. Zero gradient:
At a saddle point, the gradient of the cost function is zero. This means that the partial derivatives with respect to each parameter are all equal to zero.

2. Neither Minimum nor Maximum:
Unlike a local minimum or maximum, a saddle point is a point where the cost function neither reaches a minimum nor a maximum value.

3. Flat in Some Dimensions, Steep in Others:
The surface of the cost function is flat in certain dimensions (where the partial derivatives are zero) and steep in others. It creates a saddle-like shape.

4. Challenge for Optimization Algorithms:
Optimization algorithms, such as gradient descent, can get stuck or converge very slowly near saddle points because the gradient is zero, and the algorithm may struggle to determine the right direction to move.

Dealing with Saddle Points:

1. Higher-Order Optimization:
Using higher-order optimization methods, such as Newton's method or quasi-Newton methods, can help in navigating around saddle points more effectively.

2. Random Initialization:
Random initialization of parameters can help algorithms escape saddle points and find better solutions.

3. Noise Injection:
Injecting noise into the optimization process can help the algorithm overcome saddle points by introducing randomness.

4. Gradient Clipping:
Gradient clipping can prevent the gradients from becoming too small, allowing the optimization algorithm to make more meaningful updates.

5. Use of Momentum:
Techniques like momentum in optimization algorithms can help carry the algorithm through flat regions, reducing the chances of getting stuck at saddle points.

In Summary:
Saddle points present challenges in the optimization of cost functions in machine learning. While not as problematic as local minima, they can still hinder the convergence of optimization algorithms. Researchers and practitioners employ various strategies, such as higher-order optimization, random initialization, and noise injection, to mitigate the impact of saddle points and improve the efficiency of optimization processes in high-dimensional spaces.

Difference between convex optimization and non-convex optimization:

Convex optimization	Non-convex optimization
A *convex optimization problem* is a problem where all of the constraints are convex functions, and the objective is a convex function if minimizing, or a concave function if maximizing. Linear functions are convex, so linear programming problems are convex problems.	A *non-convex optimization problem* is any problem where the objective or any of the constraints are non-convex. Such a problem may have multiple feasible regions and multiple locally optimal points within each region.
Convex functions have a unique global minimum, making optimization easier and more reliable.	Non-convex functions, on the other hand, can have multiple local minima, making optimization more challenging.
Convex optimization problems are generally easier to solve computationally due to the properties of convex functions.	Non-convex optimization problems are more computationally challenging as they require more sophisticated algorithms to explore the solution space efficiently and avoid getting trapped in local minima.
Convex optimization algorithms are guaranteed to converge to the global minimum given certain conditions, such as convexity and continuity of the objective function.	Non-convex optimization algorithms may not always converge to the global minimum, especially when dealing with complex functions with multiple local minima.
Convex optimization is widely used in various fields such as machine learning, engineering, economics, and operations research due to its well-understood properties and efficient algorithms.	Non-convex optimization is essential in many real-world problems where the objective function is inherently non-convex, such as deep learning, molecular modeling, and financial modeling.

Advantages of RMSprop over Adagrad:

Adagrad (short for Adaptive Gradient) is an adaptive learning rate optimization algorithm that adapts the learning rate for each parameter based on the historical gradients. It is well-suited for sparse data and large-scale problems. Adagrad computes the learning rate for each parameter individually, which allows it to converge quickly on sparse features. However, Adagrad's learning rate can become too small as training progresses, leading to slow convergence.

Advantages:

- Adapts the learning rate for each parameter based on the sum of the squares of its past gradients, making it suitable for sparse datasets

- Performs well for problems with low learning rates

Disadvantages:

- The learning rate can become too small over time, hindering convergence

- Requires more memory than SGD due to the need to store past gradients' sums

RMSProp (short for Root Mean Square Propagation) is an adaptive learning rate optimization algorithm that divides the learning rate by a moving average of the square root of the accumulated gradients. It is well-suited for non-convex problems and allows the model to

adapt the learning rate to different features in the data. However, RMSProp can suffer from slow convergence in some cases.

Advantages:

- Adapts the learning rate for each parameter based on an exponentially decaying average of past squared gradients, making it suitable for deep neural networks with many layers

- Performs well in various domains

Disadvantages:

- May converge slowly compared to other optimizers like Adam

- Not suitable for problems with very sparse data

Conclusion:

- **RMSProp** adapts the learning rates based on recent gradient magnitudes, making it effective for non-stationary objectives and neural networks.
- **Adagrad** is particularly effective for sparse data, adapting the learning rate to different parameters, but it may suffer from a constantly decreasing learning rate.