3/4/25

Gradient of Vector Valued Functions
or
Jacobian matrix.

Lec 31 :-

We know the defn of partial deriv. of a scalar valued function. If $f: \mathbb{R}^n \to \mathbb{R}^m$ is a vector valued function where $n, m \geq 1$ & $x \in \mathbb{R}^n$, then $f(x)$ can be written as

$$f(x) = (f_1(x), f_2(x), \ldots, f_m(x))$$

where ~~each~~ $f_i(x)$ denotes the i^{th} component of $f(x)$ for $1 \leq i \leq m$.

$$x = (x_1, x_2, \ldots, x_n) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = [x_1 \; x_2 \ldots x_n]^T$$

we will use column vector notation for vectors, because we will deal with matrices. & $\mathbb{R}^n \cong M_{n\times 1}(\mathbb{R})$.

Further $f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix} = [f_1(x) \ldots f_m(x)]^T \in \mathbb{R}^m$.

& $f = [f_1 \; f_2 \ldots f_m]^T$

Defn (Partial deriv of ~~a~~ f)

$$\frac{\partial f}{\partial x_i} = \left[\frac{\partial f_1}{\partial x_i} \quad \frac{\partial f_2}{\partial x_i} \ldots \frac{\partial f_m}{\partial x_i}\right]^T$$

Recall chain rule. If $w = f(x_1, x_2, x_3)$ is a scalar function in $D \subseteq \mathbb{R}^3$ which is $C^1(D)$ (ie all first ~~parti~~ order partials exist in D & are cont. in D) & x_1, x_2 & x_3 are themselves functions of u_1 & u_2, ie,

$$x_1 = x_1(u_1, u_2)$$
$$x_2 = x_2(u_1, u_2)$$
$$x_3 = x_3(u_1, u_2)$$

each of which ~~satisfy~~ $x_i \in C^1(B)$ &, where B is in $u_1 u_2$ plane

where if $(u_1, u_2) \in B$, then $(x_1(u_1,u_2), x_2(u_1,u_2), x_3(u_1,u_2)) \in D$, then partial derivative of w wrt u_1 & u_2 can be written in terms of x_1, x_2 & x_3 as

$$\frac{\partial w}{\partial u_1} = \frac{\partial w}{\partial x_1}\frac{\partial x_1}{\partial u_1} + \frac{\partial w}{\partial x_2}\frac{\partial x_2}{\partial u_1} + \frac{\partial w}{\partial x_3}\frac{\partial x_3}{\partial u_1}$$

$$\& \quad \frac{\partial w}{\partial u_2} = \frac{\partial w}{\partial x_1}\frac{\partial x_1}{\partial u_2} + \cdots + \frac{\partial w}{\partial x_3}\frac{\partial x_3}{\partial u_2}$$

$$\underbrace{\begin{bmatrix} \frac{\partial w}{\partial u_1} & \frac{\partial w}{\partial u_2} \end{bmatrix}}_{\nabla_u w} = \underbrace{\begin{bmatrix} \frac{\partial w}{\partial x_1} & \frac{\partial w}{\partial x_2} & \frac{\partial w}{\partial x_3} \end{bmatrix}}_{\nabla_x w} \begin{bmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_1}{\partial u_2} \\ \frac{\partial x_2}{\partial u_1} & \frac{\partial x_2}{\partial u_2} \\ \frac{\partial x_3}{\partial u_1} & \frac{\partial x_3}{\partial u_2} \end{bmatrix}}_{h}$$

~~where $h(u_1,u_2) = (x_1(u_1,u_2), x_2(u_1,u_2), x_3(u_1,u_2))$. The mat~~

To see chain rule as matrix multiplication, it is useful to see gradient as row vectors instead of column vectors.

<u>Defn</u> (Gradient of vector valued fns)

Let $f: \mathbb{R}^n \to \mathbb{R}^m$ & $x \in \mathbb{R}^n$. Then, define

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

From defn of pd for vector valued f, we rewrite this as

$$= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

This matrix is called the gradient of f or Jacobian of f.

For $a \in \mathbb{R}^n$, $\nabla_x f(a) = \left[\frac{\partial f_i}{\partial x_j}(a) \right]_{i,j}$

Eg 1) a) find the gradient of $f(x) = Ax$ where $f: \mathbb{R}^N \to \mathbb{R}^M$ & $A \in M_{M\times N}(\mathbb{R})$.

Soln :- To compute $\nabla_x f$, we need to know $\frac{\partial f_i}{\partial x_j}$ where $f = [f_1 \dots f_M]^T$ & $x = [x_1\ x_2 \dots x_N]^T$.

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_M(x) \end{bmatrix} = Ax = \begin{bmatrix} A_{11} & A_{12} \dots A_{1N} \\ A_{21} & A_{22} \dots A_{2N} \\ \vdots & \vdots \\ A_{M1} & A_{M2} \dots A_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

$$\Rightarrow f_1(x) = A_{11}x_1 + \dots + A_{1N}x_N = \sum_{j=1}^{N} A_{1j}x_j$$

More generally, $f_i(x) = \sum_{j=1}^{N} A_{ij}x_j$

$$\therefore \frac{\partial f_i}{\partial x_j} = A_{ij} \quad \forall \quad 1 \le i \le M \ \& \ 1 \le j \le N.$$

$$\therefore \nabla_x f = \begin{bmatrix} A_{11} & A_{12} \dots A_{1N} \\ \vdots & \vdots \\ A_{M1} & A_{M2} \dots A_{MN} \end{bmatrix} = A$$

& b (constant) $\in \mathbb{R}^M$

9/4/25 ie, $\nabla_x f$ evaluated at any $a \in \mathbb{R}^N$ is A itself. $\nabla_x f = A$

Lec 32 Ex. If $f(x) = Ax + b$ where $x \in \mathbb{R}^N$; $A \in M_{M\times n}(\mathbb{R})$, show that ~~$\nabla f = A$~~

(Supervised Learning ~~Eqtns~~) – Appln in Machine Learning $\hat{y} = \Phi\theta$ to predict y

Eg 2) Let us consider the linear model ~~$\hat{y} = A\Phi\theta$~~ where θ is a parameter vector, $\theta \in \mathbb{R}^D \cong M_{D\times 1}(\mathbb{R})$ & $\Phi \in M_{N\times D}(\mathbb{R})$ are input features

Let $e(\theta) = \hat{y} - \Phi\theta$ & $L(e) = \|e\|^2$. Find ∇grad L wrt θ.

Soln :- $\hat{y} \in \mathbb{R}^N$ & hence $e(\theta) \in \mathbb{R}^N$. So, $\mathbb{R}^D \xrightarrow{e} \mathbb{R}^N$

$\theta \longrightarrow \hat{y} - \Phi\theta$.

$L(e) = \|e\|^2 \Rightarrow \quad \mathbb{R}^N \xrightarrow{L} \mathbb{R}$

$x \longrightarrow \|x\|^2.$

$\mathbb{R}^D \xrightarrow{e} \mathbb{R}^N \xrightarrow{L} \mathbb{R}$

$\theta \longrightarrow y - \Phi\theta \longrightarrow \|y - \Phi\theta\|^2$

$\nabla_\theta L = \nabla_e L \times \nabla_\theta e$

↳ Gradient of vector valued e or Jacobian

$$\nabla_\theta e = \begin{bmatrix} \frac{\partial e_1}{\partial \theta_1} & \cdots & \frac{\partial e_1}{\partial \theta_D} \\ \vdots & & \\ \frac{\partial e_N}{\partial \theta_1} & \cdots & \frac{\partial e_N}{\partial \theta_D} \end{bmatrix}$$

~~$e_1(\theta) = y^1$~~ $e(\theta) = y - \Phi\theta$ is a linear fn & hence

by prev. problem $\left(\nabla_\theta e\right)(a) = -\Phi.$

Now,

$$\nabla_e L = \begin{bmatrix} \frac{\partial L}{\partial e_1} & \cdots & \frac{\partial L}{\partial e_N} \end{bmatrix}$$

$L(e_1, \ldots, e_N) = e_1^2 + e_2^2 + \cdots + e_N^2$

$= \begin{bmatrix} 2e_1 & 2e_2 & \cdots & 2e_N \end{bmatrix}$

$= 2e^T = 2(y - \Phi\theta)^T = 2(y^T - \theta^T\Phi^T)$

$\Rightarrow \nabla_\theta L = -2 \cdot \underbrace{(y^T - \theta^T\Phi^T)}_{1\times N} \underbrace{\Phi}_{N\times D}$

Supervised learning problem

Suppose you have m many training examples say (x^i, y^i) $1 \le i \le m$ where $x^i \in \mathbb{R}^n$. Say for example

	Living area (x_1)	No of bedrooms (x_2)	...	n^{th} feature (x_n)	Price of the house (y)
1	x_1^1	x_2^1	..	x_n^1	y^1
2	x_1^2	x_2^2	..	x_n^2	y^2
⋮					
m	x_1^m	x_2^m	..	x_n^m	y^m

We want to find a nice function $h : \mathbb{R}^n \longrightarrow \mathbb{R}$ such that $h(\vec{x})$ (where $\vec{x} = (x_1, x_2, \cdots, x_n)$) is a good predictor for the corresponding value of $\vec{y}$.

These problems are called regression problems. We will consider the simplest ~~model~~ among those called linear regression problems where we approximate y using a linear function of $\vec{x}$.

ie, ~~$h(\vec{x}) =$~~ $h(x_1, x_2, \cdots, x_n) = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

where θ_i-s are called parameters or weights. Note that

$$h\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}\right) = x^T \theta$$ See x as column vectors (Δθ)

~~let h~~ x^i for each $1 \le i \le m$ is also a column vector

$$\Rightarrow h\left(\begin{bmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_n^i \end{bmatrix}\right) = (x^i)^T \theta$$

(4)

$$\begin{bmatrix} h(x^1) & h(x^2) & \cdots & h(x^m) \end{bmatrix} = \begin{bmatrix} (x^1)^T\theta & (x^2)^T\theta & \cdots & (x^m)^T\theta \end{bmatrix}$$

$$= \underline{\phi} \in n \times m \text{ matrix}$$

Here $x^1 = \begin{pmatrix} x_1^1 \\ x_2^1 \\ \vdots \\ x_n^1 \end{pmatrix}$

$y^i - h_\theta(x^i)$ will be diff btwn actual value & predicted value, ie, the error. Here $\theta \in \mathbb{R}^n$ & $x^i \in \mathbb{R}^n$

$$e(\theta) = \begin{bmatrix} y^1 - h_\theta(x^1) \\ y^2 - h_\theta(x^2) \\ \vdots \\ y^m - h_\theta(x^m) \end{bmatrix} = \begin{bmatrix} y^1 - (x^1)^T\theta \\ y^2 - (x^2)^T\theta \\ \vdots \\ y^m - (x^m)^T\theta \end{bmatrix}$$

$$\begin{bmatrix} (x^1)^T\theta \\ (x^2)^T\theta \\ \vdots \\ (x^m)^T\theta \end{bmatrix} = \underset{m \times n}{\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & & & \\ x_1^m & x_2^m & \cdots & x_n^m \end{bmatrix}} \underset{n \times 1}{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}}$$

$$\therefore \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{bmatrix} - \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & & & \\ x_1^m & x_2^m & \cdots & x_n^m \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

→ input features

or

$$e(\theta) = y - \underline{\phi}\theta$$

$\mathbb{R}^m$ → $M_{m \times n}(\mathbb{R})$ → $\mathbb{R}^n$

In such problems, we are interested in minimising the error fn $e(\theta)$ ie, $\min \| e(\theta) \|^2$ for a suitable θ. AIM: finding θ such that $J(\theta)$ is min

Gradients of Matrices

Suppose we have two matrices $A \in M_{m\times n}(\mathbb{R})$ & $B \in M_{p\times q}(\mathbb{R})$ & wish to compute the gradient of A w.r.t B, (think of B as a variable, but matrix & A is dependent on B i.e, $A = f(B)$) the resulting Jacobian

Then, $A_{11}, \dots, A_{1n}, A_{21}, \dots A_{2n}, \dots, A_{m1}, \dots A_{mn}$ are the component functions of A & each of them is a function of pq many variables. Its gradient would be the row vector

$$\nabla_B A_{11} = \begin{bmatrix} \frac{\partial A_{11}}{\partial B_{11}} & \frac{\partial A_{11}}{\partial B_{12}} \dots \frac{\partial A_{11}}{\partial B_{1q}} & \frac{\partial A_{11}}{\partial B_{21}} \dots \frac{\partial A_{11}}{\partial B_{2q}} & \dots & \frac{\partial A_{11}}{\partial B_{p1}} \dots \frac{\partial A_{11}}{\partial B_{pq}} \end{bmatrix}$$

Sometimes it is also written as matrix

$$\begin{bmatrix} \frac{\partial A_{11}}{\partial B_{11}} & \frac{\partial A_{11}}{\partial B_{12}} \dots & \frac{\partial A_{11}}{\partial B_{1q}} \\ \frac{\partial A_{11}}{\partial B_{21}} & \dots & \frac{\partial A_{11}}{\partial B_{2q}} \\ \vdots & & \\ \frac{\partial A_{11}}{\partial B_{p1}} & & \frac{\partial A_{11}}{\partial B_{pq}} \end{bmatrix}$$

Each component's gradient is a $p\times q$ matrix & since there are mn component functions, the Jacobian would be $m\times n \times p\times q$ matrix, ie, a four-dimensional tensor J whose entries are given as $J_{ijkl} = \frac{\partial A_{ij}}{\partial B_{kl}}$.

This form of representation of gradient of matrices is highly inconvenient & since $M_{m\times n}(\mathbb{R}) \cong \mathbb{R}^{mn}$, we see that f is equivalently a function as below :-

$$\mathbb{R}^{pq} \xrightarrow{f} \mathbb{R}^{mn}$$

ie, Matrix is reshaped into vector
(B) $(\tilde{B} \in \mathbb{R}^{pq})$

Eg 1) (Gradient of vectors with respect to matrices)

Consider the map $A \xrightarrow{f} Ax$ $(Ax \in \mathbb{R}^M)$

where $A \in M \times N$ matrix & $x \in \mathbb{R}^N$

Ax – vector
A – matrix

i) What is the dimension of $\nabla_A f$?

2) Compute the gradient of f w.r.t A.

Soln :- Since $Ax \in M_{M\times 1}(\mathbb{R})$, $f : \mathbb{R}^{MN} \longrightarrow \mathbb{R}^M$

~~Each component function f_i~~ & Hence $\nabla_A f \in M_{M\times(M\times N)}$

2) Let $f_1(A), f_2(A), \ldots, f_M(A)$ be its component functions.

What is $\dfrac{\partial f_i}{\partial A_{kj}}$? where $1 \le i \le M$, $1 \le k \le M$

$1 \le j \le N$

Let $x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$.

$Ax \in \mathbb{R}^M$. i^{th} entry of Ax, ie, $(Ax)_i$ is given by

$$Ax = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ A_{i1} & A_{i2} & \cdots & A_{iN} \\ \vdots & & & \\ A_{M1} & A_{M2} & \cdots & A_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{bmatrix}$$

$$(Ax)_i = [A_{i1}\ A_{i2} \cdots A_{iN}] \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \sum_{k=1}^{N} A_{ik} x_k$$

$$\therefore \frac{\partial (f_i)}{\partial A_{kj}} = \frac{\partial (Ax)_i}{\partial A_{kj}} = \begin{cases} x_j & \text{if } k = i \\ 0 & \text{if } k \neq i \end{cases}$$

$$\therefore \nabla_A f = \begin{bmatrix} \frac{\partial f_1}{\partial A_{11}} & \frac{\partial f_1}{\partial A_{12}} \cdots \frac{\partial f_1}{\partial A_{1N}} & \frac{\partial f_1}{\partial A_{21}} \cdots \frac{\partial f_1}{\partial A_{2N}} & \cdots & \frac{\partial f_1}{\partial A_{M1}} \cdots \frac{\partial f_1}{\partial A_{MN}} \\ \vdots & & & & \\ \frac{\partial f_M}{\partial A_{11}} \cdots \frac{\partial f_M}{\partial A_{1N}} & \frac{\partial f_M}{\partial A_{21}} \cdots \frac{\partial f_M}{\partial A_{2N}} & \cdots & \frac{\partial f_M}{\partial A_{M1}} \cdots \frac{\partial f_M}{\partial A_{MN}} \end{bmatrix}_{M \times (M \times N)}$$

$$= \begin{bmatrix} x_1 \; x_2 \cdots x_N & 0 \cdots 0 & 0 \cdots 0 & 0 \cdots 0 \\ 0 & x_1 \; x_2 \cdots x_N & 0 & 0 \\ & & \ddots & \\ & & & x_1 \; x_2 \cdots x_N \end{bmatrix}$$

Follow (Diesenroth : Mathematics for Machine Learning. Faisal Cheng Soon Ong)

Lec 33 :- Eg 2) (Gradient of Matrices w.r.t matrices)

10/4/25

Let $A \in M_{M \times N}(\mathbb{R})$ & $f(A) = A^T A = K$

ie, $f : M_{M \times N}(\mathbb{R}) \longrightarrow M_{N \times N}(\mathbb{R})$.

What is $\nabla_A K$?

Soln :- Let A_i denote i^{th} row of A & A_{ij} denote ij^{th} entry of A.

Then $K_{pq} = (A^T A)_{pq}$

$= (p^{th} \text{ row of } A^T) \cdot (q^{th} \text{ col. of } A) = (p^{th} \text{ col of } A) \cdot (q^{th} \text{ col. of } A)$

$= A_p \cdot A_q = \sum_{m=1}^{M} A_{mp} A_{mq}$

$$A = \begin{pmatrix} A_{1p} & & A_{1N} \\ A_{2p} & \cdots & \\ \vdots & & \\ A_{Mp} & & A_{MN} \end{pmatrix}$$

$q = p \Rightarrow K_{pp} = \sum_{m=1}^{M} (A_{mp})^2$.

$$\therefore \frac{\partial K_{pq}}{\partial A_{ij}} = \begin{cases} 2A_{ip} & p = q \;\&\; j = p \\ \bullet & \\ A_{iq} & p \neq q \text{ but } j = p \\ A_{ip} & p \neq q \text{ but } j = q \\ 0 & j \notin \{p, q\} \end{cases}$$

$\nabla_A f \in \mathbb{R}^{(N \times N) \times (M \times N)}$

Suppose I ~~Here, we will~~ follow the convention that

$$f_1 = K_{11}$$
$$f_2 = K_{21}$$
$$\vdots$$
$$f_N = K_{N1}$$
$$\vdots$$
$$f_{N^2} = K_{NN}$$

ie, component functions are ordered along the columns first & then rows.

Eg. If $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix}_{3\times 2}$

Then $K = A^T A = \begin{bmatrix} (A_{11})^2 + (A_{21})^2 + (A_{31})^2 = f_1 & A_{11}A_{12} + A_{21}A_{22} + A_{31}A_{32} = f_3 \\ A_{12}A_{11} + A_{22}A_{21} + A_{32}A_{31} = f_2 & (A_{12})^2 + (A_{22})^2 + (A_{32})^2 = f_4 \end{bmatrix}$

$$\nabla_A f_1 = \begin{bmatrix} \frac{\partial f_1}{\partial A_{11}} & \cdots & \frac{\partial f_1}{\partial A_{32}} \end{bmatrix} = \begin{bmatrix} 2A_{11} & 0 & 2A_{21} & 0 & 2A_{31} & 0 \end{bmatrix}$$

$$\nabla_A f_2 = \nabla_A f_3 = \begin{bmatrix} A_{12} & A_{11} & A_{22} & A_{21} & A_{32} & A_{31} \end{bmatrix}$$

$$\vdots$$

$$\nabla_A f_4 = \begin{bmatrix} 0 & 2A_{12} & 0 & 2A_{22} & 0 & 2A_{32} \end{bmatrix}$$

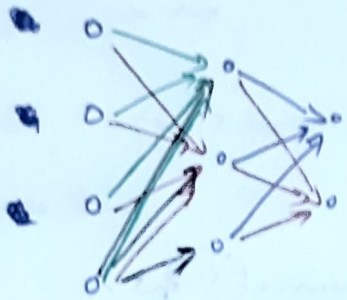
~~$\nabla_A f \in \mathbb{R}^{(N\times N)\times(M\times N)}$~~

$$\nabla_A f = \begin{bmatrix} 2A_{11} & 0 & 2A_{21} & 0 & 2A_{31} & 0 \\ A_{12} & A_{11} & A_{22} & A_{21} & A_{32} & A_{31} \\ A_{12} & A_{11} & A_{22} & A_{21} & A_{32} & A_{31} \\ 0 & 2A_{12} & 0 & 2A_{22} & 0 & 2A_{32} \end{bmatrix}_{(2\times 2)\times(3\times 2)}$$

(Lec 33 ctd)

Artificial Neural Network

Anyone familiar with neural networks would have seen the full diagram.



Here, circles denote neurons & lines correspond to the network of connection between them.

Neural network is fundamentally a mathematical function that takes an input (which could be vector) x & gives an output y (vector)

Case 1 :-

$$a^{(0)} \longrightarrow a^{(1)}$$

Define $a^{(1)} = \sigma(wa^{(0)} + b)$ where $b, w \in \mathbb{R}$

σ is a function called activation function.

a activity

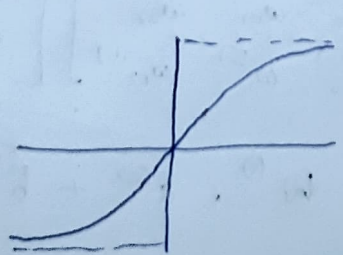
w weight

b bias

(When the stimulation of one or more neurons goes beyond a threshold, the neuron on which they interacts gets "activated".

An example of a function that has this thresholding property is the hyperbolic tangent $(\text{Tanh})(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

Note : $\lim_{x \to \infty} \text{Tanh}\, x = 1$ & $\lim_{x \to -\infty} \text{Tanh}\, x = -1$

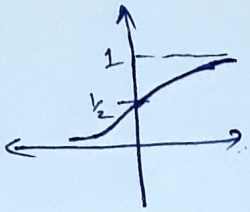


which have this thresholding property

Class of such functions ~~in general~~ are called sigmoids

Eg. $f(x) = \frac{1}{1+e^{-x}}$ Lt $f(x) = 1$ as $x \to \infty$

Lt $f(x) = 0$ as $x \to -\infty$

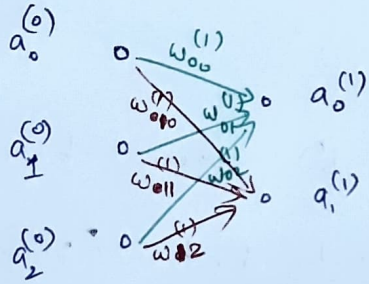


Case 2 :- (Input) 2 neurons achraing 1 neuron (Output)

b^1 – bias of neuron 1

$$a^{(1)} = \sigma(w_0 a_0^0 + w_1 a_1^0 + b^1)$$

Case 3 :- Input 3 neurons & Output 2 neurons



$w_{ij}^{(1)}$ is the weight / line connecting neuron a_i of 1st layer (ie, $a_i^{(1)}$) & a_j of 0^{th} layer (ie, $a_j^{(0)}$).

$a_j^{(0)} \xrightarrow{w_{ij}^{(1)}} a_i^{(1)}$

- $b_0^{(1)}$ is the bias of neuron a_0 in 1st layer.

$b_1^{(1)}$ (—— " ——) a_1 (— " —>

$$a_0^{(1)} = \sigma(w_{00}^{(1)} a_0^{(0)} + w_{01}^{(1)} a_1^{(0)} + w_{02}^{(1)} a_2^{(0)} + b_0^{(1)})$$

$$a_1^{(1)} = \sigma(w_{10}^{(1)} a_0^{(0)} + w_{11}^{(1)} a_1^{(0)} + w_{12} a_2^{(0)} + b_1^{(1)})$$

$$\vec{a^{(1)}} = \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{00}^{(1)} & w_{01}^{(1)} & w_{02}^{(1)} \\ w_{10}^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ a_2^{(0)} \end{bmatrix} + \begin{bmatrix} b_0^{(1)} \\ b_1^{(1)} \end{bmatrix} \right)$$

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

This is an example of a single layered neural network with 3 inputs & 2 output & can be represented as

$$a^{(1)} = \sigma\left(W^{(1)} \cdot a^{(0)} + b^{(1)} \right).$$

More generally, when there are n inputs & m outputs, we can represent it is

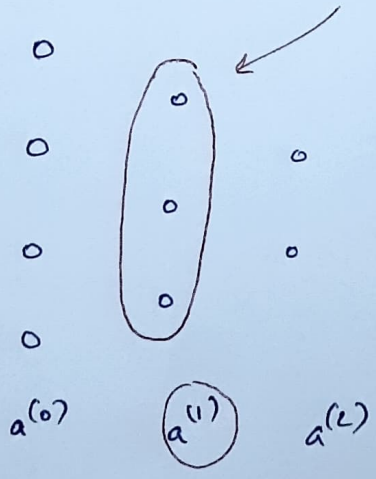
$$a^{(1)} = \sigma\left(W^{(1)} \cdot a^{(0)} + b^{(1)} \right) \quad \text{where}$$

$$W^{(1)} = \begin{bmatrix} w_{00}^{(1)} & w_{01}^{(1)} & \cdots & w_{0,n-1}^{(1)} \\ w_{10}^{(1)} & w_{11}^{(1)} & \cdots & w_{1,n-1}^{(1)} \\ \vdots & \vdots & & \vdots \\ w_{m-1,0}^{(1)} & w_{m-1,1}^{(1)} & \cdots & w_{m-1,n-1}^{(1)} \end{bmatrix}$$

$$\vec{a^{(0)}} = \begin{bmatrix} a_0^{(0)} \\ \vdots \\ a_{n-1}^{(0)} \end{bmatrix} \quad \& \quad \vec{b^{(1)}} = \begin{bmatrix} b_0^{(1)} \\ b_1^{(1)} \\ \vdots \\ b_{m-1}^{(1)} \end{bmatrix}$$

Neural networks may have ~~more than one~~ one or more layers of neurons btwn the input & the output. These are called hidden layers. (The outputs of a ~~hidden~~ layer here become inputs for the next layer.)

Neurons in layer (1) becomes inputs for neurons in layer (2).



$$\vec{a^{(1)}} = \sigma\left(W^{(1)} \vec{a^{(0)}} + \vec{b^{(1)}} \right)$$

$$\vdots$$

$$\vec{a^{(2)}} = \sigma\left(W^{(2)} \vec{a^{(1)}} + \vec{b^{(2)}} \right).$$

More generally,

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

Such neural networks are sometimes called "feed forward" neural networks.

Persuading a neural network to do tasks (like image recognition etc) now becomes just a job of teaching it the right weights & biases.

Eg 1)

To train the network to give a not function

(ie, if you input 1, it returns 0

if you input 0, it returns 1.)

Let us use $\sigma(x) = \tanh x$.

$$a^{(1)} = \sigma\left(a^{(0)} w^{(1)} + b^{(1)}\right) \qquad w^{1} -$$

Which among the following weight & basis gives the best result ?

a) $w^{(1)} = 10, \quad b^{(1)} = 0$

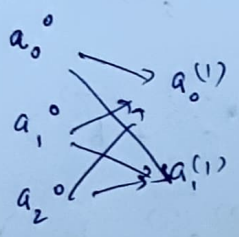
b) $w^{(1)} = -3, \quad b^{(1)} = 0$

c) $w^{(1)} = 3, \quad b^{(1)} = 1$

✓ d) $w^{(1)} = -5, \quad b^{(1)} = 5$

e) $w^{(1)} = 0, \quad b^{(1)} = 5$

Eg 2) Consider



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

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

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

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

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

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

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