X-OR problem (contd.) Formulate this as a regression problem and use MSE loss (Mean Square Error)

Training set $\underline{x} = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}$ $\quad X = [\underline{x}_{(1)} \; \underline{x}_{(2)} \; \underline{x}_{(3)} \; \underline{x}_{(4)}] = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

$$\underline{t} = [t_{(1)} \; t_{(2)} \; t_{(3)} \; t_{(4)}] = [0 \; 1 \; 1 \; 0]$$

MSE loss $J(\underline{w}) \triangleq \frac{1}{4} \sum_{n=1}^{4} [y(\underline{x}_{(n)}, \underset{\text{parameters}}{\underline{w}}) - t_{(n)}]^2$

linear model

$$y(\underline{x}) = y(\underline{x}, \underline{w}) = \underbrace{\underline{w}^T \underline{x}}_{\text{homogeneous}} = \underbrace{\underline{w}^T \underline{x} + b \text{ or } w_0}_{\text{non-homogeneous}}$$

$$(*) \quad \frac{\partial J(\underline{w})}{\partial \underline{w}} = \frac{1}{4} \cdot 2 \sum_{(n)} \{ y(\underline{x}_{(n)}, \underline{w}) - t_{(n)} \} \frac{\partial y}{\partial \underline{w}} (\underline{x}_{(n)}, \underline{w}) = 0$$

$$\Rightarrow \frac{\partial (\underline{w}^T \underline{x})}{\partial \underline{w}} = \underline{x} = \frac{\partial (\underline{x}^T \underline{w})}{\partial \underline{w}} \quad \text{(Rule)}$$

$$\Rightarrow \boxed{\frac{1}{2} \sum_{(n)} \{ y(\underline{x}_{(n)}, \underline{w}) - t_{(n)} \} \underline{x}_{(n)} = \underline{0}} \quad (1)$$

$$(*) \quad \frac{\partial J(\underline{w})}{\partial b} = \frac{1}{4} \cdot 2 \sum_{(n)} \{ y(\underline{x}_{(n)}, \underline{w}) - t_{(n)} \} \cdot 1 = 0$$

$$\frac{\partial}{\partial b} (\underline{w}^T \underline{x} + b) = 1$$

$$\Rightarrow \boxed{\sum_{(n)} \{ \underline{w}^T \underline{x}_{(n)} + b - t_{(n)} \} = 0} \quad (2)$$

$$① \Rightarrow \{w_2(0) + w_1(0) + b - 0\}\begin{bmatrix}0\\0\end{bmatrix}$$
$$+ \{w_2(0) + w_1(1) + b - 1\}\begin{bmatrix}0\\1\end{bmatrix} \quad = \begin{bmatrix}0\\0\end{bmatrix}$$
$$+ \{w_2(1) + w_1(0) + b - 1\}\begin{bmatrix}1\\0\end{bmatrix}$$
$$+ \{w_2(1) + w_1(1) + b - 0\}\begin{bmatrix}1\\1\end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} w_2 + b - 1 + w_2 + w_1 + b \\ w_1 + b - 1 + w_2 + w_1 + b \end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 2w_2 + w_1 + 2b \\ 2w_1 + w_2 + 2b \end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix} \Rightarrow \begin{cases} 2w_2 + w_1 + 2b = 1 \\ 2w_1 + w_2 + 2b = 1 \end{cases}$$

$$\leftarrow ①$$

$$② \Rightarrow w_2(0) + w_1(0) + b - 0$$
$$+ w_2(0) + w_1(1) + b - 1$$
$$+ w_2(1) + w_1(0) + b - 1$$
$$+ w_2(1) + w_1(1) + b - 0 = 0$$

$$\Rightarrow 4b + 2(w_2 + w_1) = 2 \Rightarrow 2b + (w_2 + w_1) = 1 \leftarrow ②$$

Put ② in ① to get

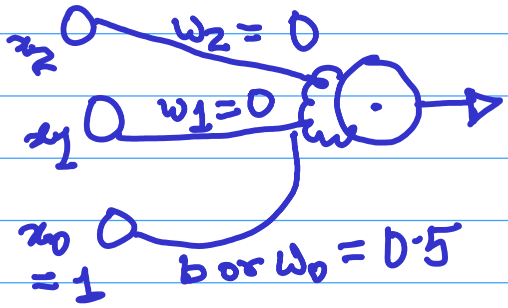
$$\underbrace{\begin{matrix} 2w_2 + \cancel{w_1} + \cancel{1} - (w_2 + \cancel{w_1}) = \cancel{1} \\ 2w_1 + \cancel{w_2} + \cancel{1} - (\cancel{w_2} + w_1) = \cancel{1} \end{matrix}} \Rightarrow \begin{matrix} w_2 = 0 \\ w_1 = 0 \\ b = 0.5 \end{matrix}$$

$$\underline{w} = \underline{0}, \quad b = 0.5$$

$$y_{(n)} = \underline{w}^T \underline{x}_{(n)} + 0.5$$

whatever is $\underline{x}$, the output should be 0.5

What did we try?



$y = \sigma(a) = a$ unit function

$y = \underline{w}^T \underline{x} + b$ (non-homogeneous)

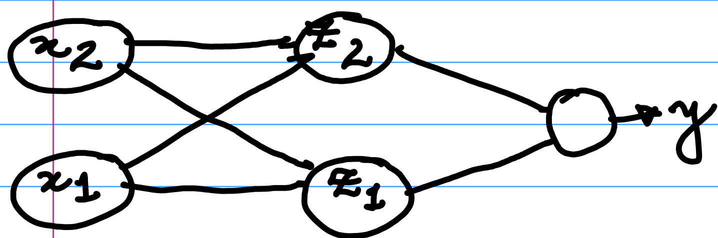
$= w_2 x_2 + w_1 x_1 + b$

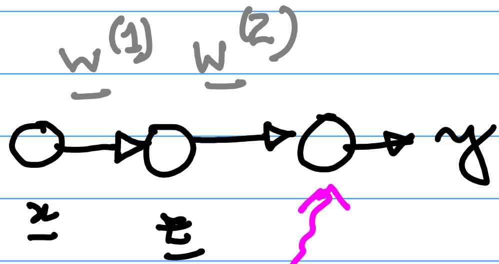
$= 0.5$ always.

Some points:→

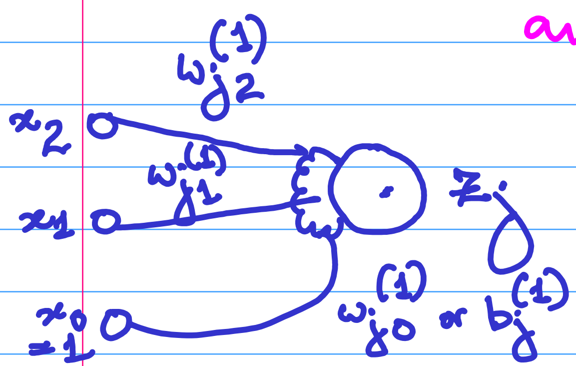
- This is a simple example with regression (though the problem is actually one of classification)
- This is one of the myriad possible solutions
- Unnecessarily ~~trying~~ to build a classifier out of a regressor (that too, for binary inputs)
- Handcrafted

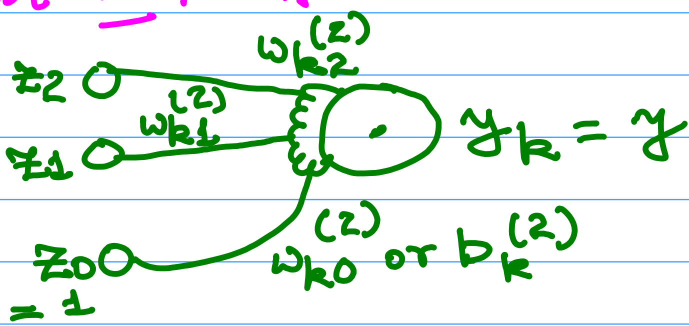
ONE possible solution :-





regressor applied to $\underline{z} = \underline{\phi}(\underline{x})$, and not $\underline{x}$ itself





Building Block (Input: 2-D: Image)

MNIST Numeral database: 28×28 images

(greyscale: not binary (not 0 and 255, or normalised 0 and 1))

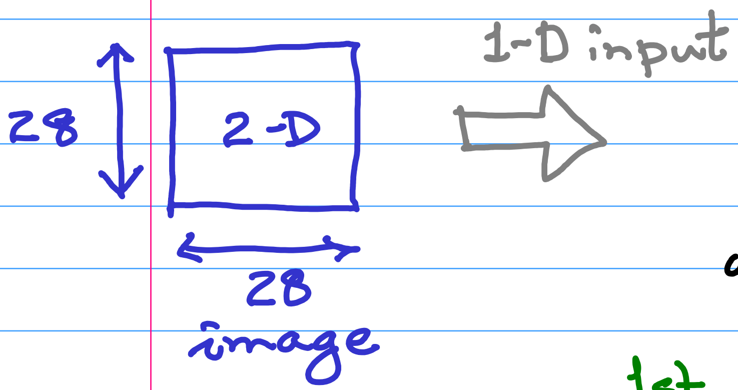
~smooth

→ shades of grey as well, though most of the image is black (0) or white (255, or 1, normalised).

Images of the 10 numerals: 0 to 9

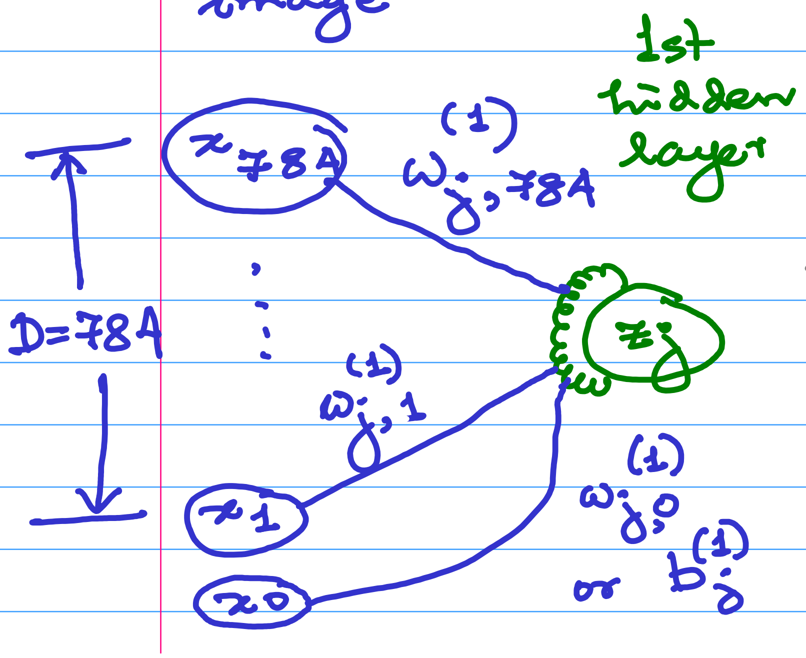
Basic structure: an MLP with 2 hidden layers

→ of specific interest is the first layer of connections

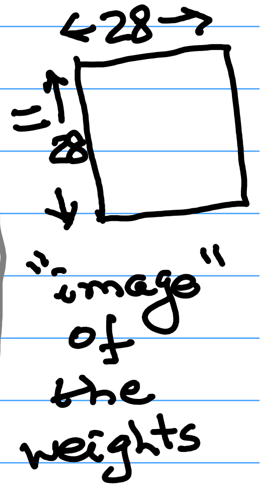


784 × 1 input vector

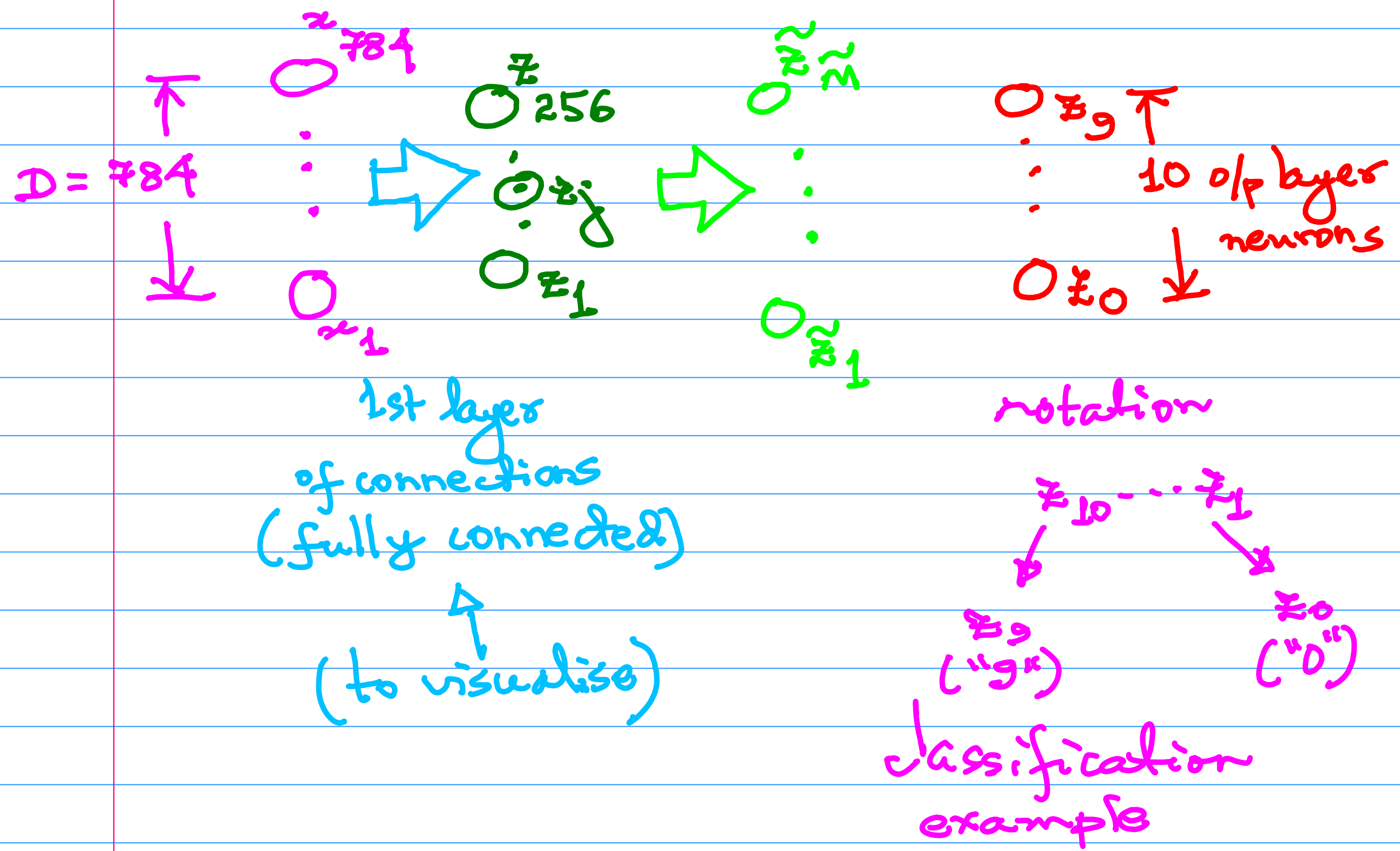
$w_{ji}^{(1)} \in \mathbb{R}$, a real number



$$\underset{784\times1}{\underline{w}^{(1)}} = \begin{bmatrix} w_{j,784}^{(1)} \\ \vdots \\ w_{j,1}^{(1)} \end{bmatrix}_{784\times1}$$

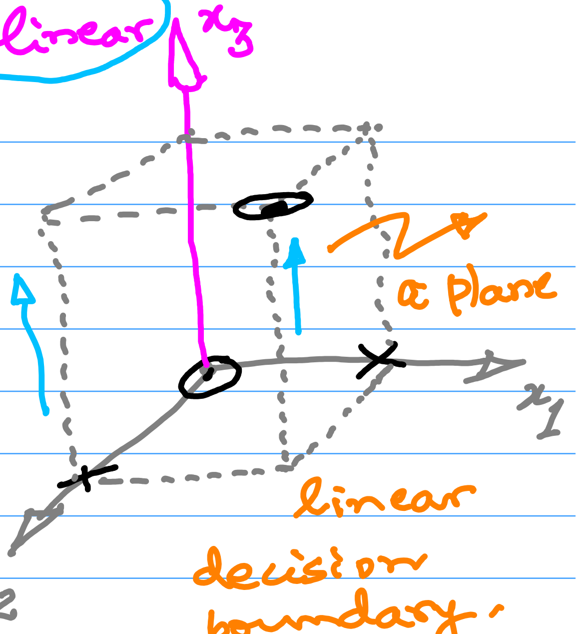


Take-home point: The input is not an ordinary 1-D vector, but a 2-D image $\Rightarrow$ we can visualise the weights not as an ordinary 1-D vector, but a 2-D image with a similar spatial configuration/arrangement as the input itself.



original features		new feature	
x_2	x_1	$x_3 = x_2 x_1$	$y = x_2 \oplus x_1$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

typically non-linear



The earlier (1, 1) point now "floats up", leading to an infinite number of planes (linear decision boundaries in 3-D) now separating the two classes (much like the concentric circles doughnut 'floating up' over the vada)

The 'Factorisation' in Math / Summation

Where does this appear, and why? Short Answer: Everywhere!

Working Rule:-
try putting it everywhere, and remove it if it is not required!

- → Probability (conditional / non-conditional)
- → Chain Rule (Calculus)
  - total derivate / partial derivative.

PROBABILITY

The general probability factorisation

$$\text{cause } 1 \rightarrow \text{Effect}$$
$$\vdots$$
$$\text{cause } N \rightarrow \text{Effect}$$

Bayes Rule

$$p(\text{cause}\#i \mid \text{effect}) = \frac{p(\text{effect} \mid \text{cause}\#i)\, p(\text{cause}\#i)}{p(\text{effect})}$$

Symmetrical

$$\underbrace{p(A|B)\,p(B)}_{p(A \text{ and } B)} = \underbrace{p(B|A)\,p(A)}_{p(B \text{ and } A)}$$

$$\boxed{p(A|B)} = \frac{p(B|A)\boxed{p(A)}}{p(B)}$$

a priori probability of A

a posteriori probability of A

updated probability of A

$$x = x + 1$$

$$\Rightarrow x_{new} = x_{old} + 1$$

$$p(A)_{updated} = [\quad] \times p(A)_{initial}$$

$p(A)$

$p(A|B)$

→ As such, there is no difference between a conditional probability and an unconditional probability → these are just the updated and initial variants of the same physical quantity:

$$p(A|B) = \frac{p(B|A)\,p(A)}{p(B)}$$

$$\underbrace{\boxed{p(A|B,C)}}_{\text{updated prob of A}} = \frac{p(B|A,C)\,\underbrace{\boxed{p(A|C)}}_{\text{initial prob of A}}}{p(B|C)}$$

$\underset{\text{updated}}{p(A)}$ ← $\boxed{\text{Bayes Rule}}$ ← $\underset{\text{initial}}{p(A)}$

Where do the summations come in

$$p(\text{effect}) = \sum_{\forall j} p(\text{effect}\,|\,\text{cause}\#j)\; p(\text{cause}\#j)$$

$\forall j$ → safely put a summation for all j's

Example: →

tap on → lawn wet ← rain

Suppose we find the lawn to be wet in the morning.

[Bayes Rule] $p\left(\begin{array}{l}\text{rained}\\ \text{last night}\end{array} \middle| \begin{array}{l}\text{lawn wet}\\ \text{in the morning}\end{array}\right)$

$$p(\text{rain} \mid \text{wet}) = \frac{p(\text{wet} \mid \text{rain})\, p(\text{rain})}{p(\text{wet})}$$

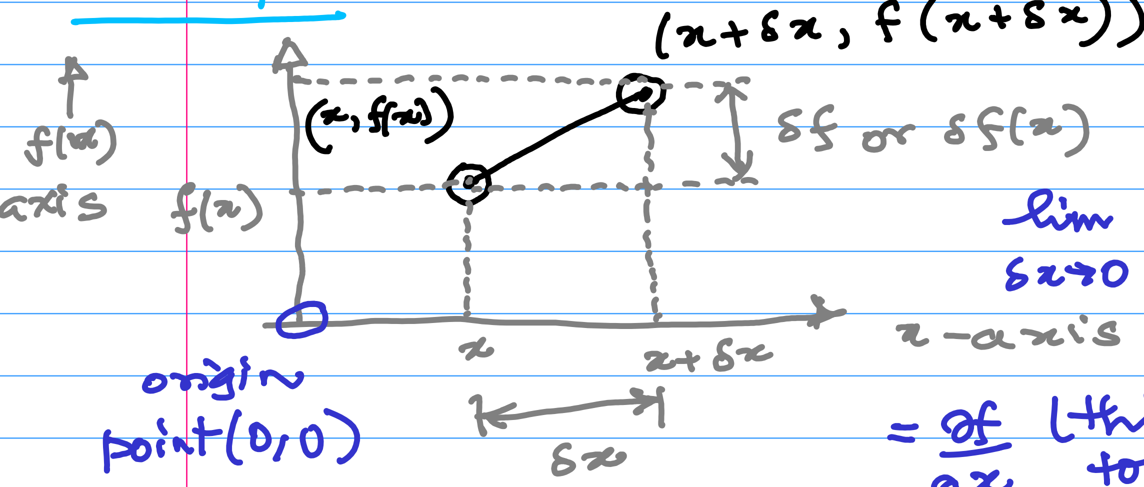
$$= \frac{p(\text{wet} \mid \text{rain})\, p(\text{rain})}{p(\text{wet} \mid \text{rain})\, p(\text{rain}) + p(\text{wet} \mid \text{tap on})\, p(\text{tap on})}$$

DERIVATIVES

The most general derivative is NOT the total derivative, but the partial derivatives.

$\boxed{1\text{-D}}$ one independent variable x → scalar
one dependent variable $f(x)$ → scalar function

Example: e.g., audio signal $f(t)$



$$\lim_{\delta x \to 0} \frac{f(x+\delta x) - f(x)}{(x+\delta x) - (x)}$$

$= \frac{\partial f}{\partial x}$ (this is also the total derivative df/dx)

Why? 1 scalar variable.

$$\lim_{\delta x \to 0} : \underbrace{f(x+\delta x) - f(x)}_{\delta f \text{ or } \delta f(x)} = \left(\frac{\partial f}{\partial x}\right) \underbrace{\delta x}_{}$$

δf or $\delta f(x)$: small change in the dependent variable, or the function

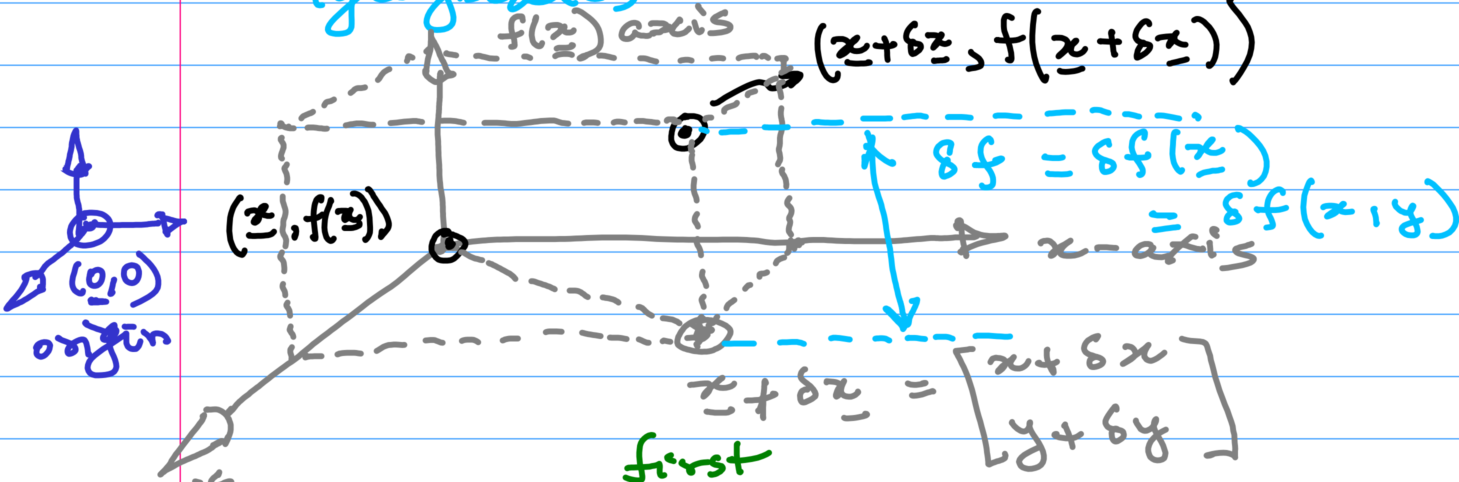
δx: small change in the independent variable.

$\boxed{2\text{-D}}$ two independent variables $\underline{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ → vector

one dependent variable $f(\underline{x})$ → scalar.

Example $I(x, y)$ or $I(\underline{x})$

the intensity of a pixel at position $\underline{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ (greyscale)



Use the first order Taylor series expansion

$$f(\underline{x}+\delta\underline{x}) = f(\underline{x}) + \vec{\nabla}_f \cdot \delta\underline{x}$$

$$\underbrace{f(\underline{x}+\delta\underline{x}) - f(\underline{x})}_{\delta f \text{ or } \delta f(\underline{x}) \text{ or } \delta f(x,y)} = \underbrace{\vec{\nabla} f \cdot \delta\underline{x}}_{}$$

$$\underbrace{\begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix} \cdot \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}}$$

$$\underbrace{\frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial y}\delta y}$$

$$= \sum_{var:\, x,y} \left(\frac{\partial f}{\partial\, var}\right)(\delta\, var)$$

$\boxed{3\text{-}D}$ Three independent variables $\underline{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ → vector

one dependent variable $f(\underline{x})$ → scalar

Example: video (greyscale) $I(x, y, t)$: Intensity at pixel (x, y) in frame # t.

Impossible to visualise $I(x, y, t) \rightarrow$ 4-D entity

Use the first order Taylor series approximation:

$$f(\underline{x}+\delta\underline{x}) = f(\underline{x}) + \vec{\nabla} f \cdot \delta\underline{x}$$

$$\underbrace{f(\underline{x}+\delta\underline{x}) - f(\underline{x})}_{\substack{\delta f \text{ or } \delta f(\underline{x}) \\ \text{or, } \delta f(x,y,z)}} = \underbrace{\begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \\ \partial f/\partial z \end{bmatrix} \cdot \begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix}}$$

$$= \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial y}\delta y + \frac{\partial f}{\partial z}\delta z$$

$$= \sum_{var = x,y,z} \left(\frac{\partial f}{\partial var}\right)(\delta var)$$

Generalise to a function of D variables

$f(\underline{x})$, $\underline{x} = \begin{bmatrix} x_D \\ \vdots \\ x_1 \end{bmatrix}$ 1st order Taylor series expansion

$$\left.\begin{matrix} \delta f \text{ or } \delta f(\underline{x}) \\ \text{or, } \delta f(x_1 \ldots x_D) \end{matrix}\right\} = \vec{\nabla} f \cdot \delta\underline{x}$$

$$= \sum_{i=1}^{D} \frac{\partial f}{\partial x_i}\delta x_i$$

Take-home point: The total change is always a summation

$$\delta f = \delta f(\underline{x}) = \sum_{i=1}^{D} \frac{\partial f}{\partial x_i}\delta x_i$$

Consider another variable t

(all the x_i's are functions of this variable t)

$$\frac{\delta f}{\delta t} = \sum_{i=1}^{D} \frac{\partial f}{\partial x_i} \frac{\delta x_i}{\delta t}$$

We can take the limit as $\delta t \to 0$

$$\frac{\partial f}{\partial t} = \sum_{i=1}^{D} \frac{\partial f}{\partial x_i} \frac{\partial x_i}{\partial t}$$

Here, we had one variable t, so the partial derivative is also the total derivative.

$$\frac{df}{dt} = \sum_{i=1}^{D} \frac{\partial f}{\partial x_i} \frac{dx_i}{dt}$$

Now, consider a set of variables t_j : $j \in \{1, \mathcal{D}\}$

(all the x_i's are functions of t_j)

$i \in \{1, D\}$ $j \in \{1, \mathcal{D}\}$

$$\boxed{\forall j : \quad \frac{\partial f}{\partial t_j} = \sum_{i=1}^{D} \frac{\partial f}{\partial x_i} \frac{\partial x_i}{\partial t_j}}$$

Chain Rule

Compact Moral of the story: if f depends on many x_i, then

$$\frac{\partial f}{\partial t} = \sum_{\forall i} \frac{\partial f}{\partial x_i} \frac{\partial x_i}{\partial t}$$

BACKPROPAGATION

1849, Cauchy

$$\underline{w}^{(\tau+1)} = \underline{w}^{(\tau)} - \eta \nabla E$$

∇E → Error function. What is this?

Gradient Descent

$$\begin{bmatrix} \vdots \\ w_{ji}^{(1)} \\ \vdots \\ w_{kj}^{(2)} \\ \vdots \end{bmatrix}$$

Training: (supervised)

[training] inputs → outputs

To learn the weights ≡ function which the NN implements

We typically start with random weights

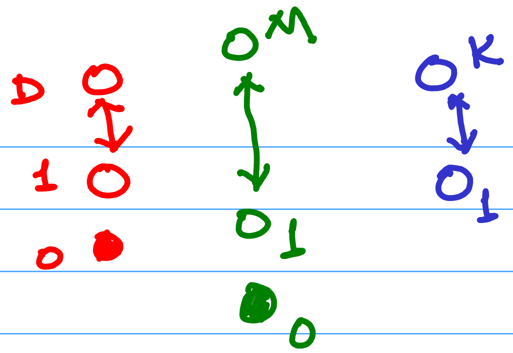
Error: b/w the ideal output & the one which the NN gives us currently.

N points
index $n = 1..N$

$$E(\underline{w}) = \sum_{n=1}^{N} E_n(\underline{w})$$

$$\frac{\partial E(\underline{w})}{\partial w} = \sum_n \partial E_n / \partial w$$

Example:



Assumptions: sum-of-squares errors

→ Hidden layer activation function

$$h(a) = \tanh(a)$$

→ Output layer activation function

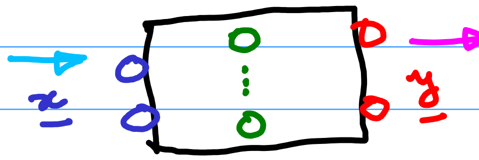
$$\sigma(a) = a, \quad y_k = a_k$$

$$h(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

$$h'(a) = \frac{\partial h(a)}{\partial a} = \frac{(e^a + e^{-a})(e^a - (-e^{-a})) - (e^a - e^{-a})(e^a - e^{-a})}{(e^a + e^{-a})^2}$$

$$= \frac{(e^a + e^{-a})^2 - (e^a - e^{-a})^2}{(e^a + e^{-a})^2} = 1 - \left[\frac{e^a - e^{-a}}{e^a + e^{-a}}\right]^2 = 1 - h^2(a)$$

For the n'th training data item:



$$E_n \triangleq \frac{1}{2} \sum_{k=1}^{K} (y_k - t_k)^2$$

→ ground truth / label

→ why only y_k? (defined only at the output layer)

y_k = k'th item of the function $\underline{y}(\underline{w}, \underline{x})$

t_k = k'th item of the target vector