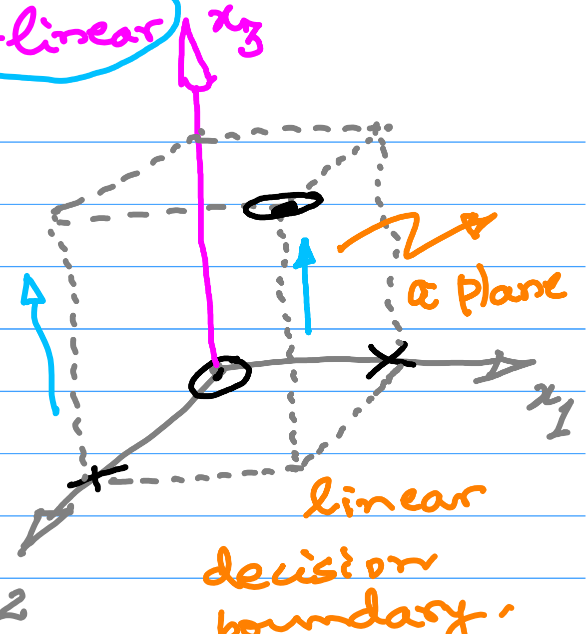
original features		new feature	$y =$
x_2	x_1	$x_3 = x_2 x_1$	$x_2 \oplus x_1$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

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

$$\begin{array}{c} \text{cause } 1 \searrow \\ \vdots \qquad \text{Effect} \\ \text{cause } N \nearrow \end{array}$$

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

$\underbrace{\text{a posteriori probability of A}}_{\text{updated probability of A}}$

$$x = x + 1$$

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

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

$p(A)$

$p(A|B)$

→ 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:

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

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

updated prob of A

initial prob of A

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

Where do the summations come in

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

→ safely put a summation for all j's

Example: →

tap on → lawn wet

rain → lawn wet

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

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

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

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

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

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

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

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 : $j \in \{1, D\}$

(all the x_i's are functions of t_j

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

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

Chain Rule

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

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

Perceptron & MLP: some closing notes

(*) Key difference: MLP and the perceptron: MLP uses continuous sigmoidal non-linearities in the hidden units, whereas the perceptron uses a step function non-linearities

(*) Variants: Skip layers: either direct connection, or with a small first layer weight (so that over its operating range, the hidden unit is effectively linear), compensating with a large weight value from the hidden unit to the output

(*) Sparse network (CNN)

Math overview/recap: → 1st order Taylor series approximation

$$\underbrace{E(\underline{w}+\delta\underline{w})}_{\text{minimise}} = E(\underline{w}) + (\vec{\nabla}E)\cdot(\delta\underline{w})$$

error function

weights (All)

Overall aim: to find a weight vector, which minimises an error function $E(\underline{w})$

scalar

vector

At the extremum, $\vec{\nabla}E = \underline{0}$ $\begin{bmatrix} \partial E/\partial w_2 \\ \partial E/\partial w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

max/min/saddle point

vector

(2-D)

Local Quadratic Approximation

$$E(\underline{w}+\delta\underline{w}) = E(\underline{w}) + \underbrace{\nabla E\cdot(\delta\underline{w})}_{(\delta\underline{w})^T\nabla E} + \frac{1}{2}(\delta\underline{w})^T H\,(\delta\underline{w}) + \underbrace{\ldots}_{\text{higher order terms}}$$

H ↳ Hessian

Example: first order

$$I(x+p,\ y+q,\ t+\tau) = I(x,y,t) + p\,I_x + q\,I_y + \tau\,I_t$$

Second order term

$$H_{ij} = \left.\frac{\partial^2 E}{\partial w_i\,\partial w_j}\right|_{\underline{w}}$$

$$E(\underline{w}+\delta\underline{w}) = E(\underline{w}) + (\delta\underline{w})^T\underbrace{\nabla E}_{=\underline{0}\text{, at the extremum}} + \frac{1}{2}(\delta\underline{w})^T H\,(\delta\underline{w})$$

Extremum:

$$E(\underline{w}+\delta\underline{w}) = E(\underline{w}) + \frac{1}{2}(\delta\underline{w})^T H\,(\delta\underline{w})$$

→ a geometric interpretation

BACKPROPAGATION

1849, Cauchy

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

Error function

What is this?

Gradient Descent

We typically start with random weights

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

Training: (Supervised)

[training] inputs $\longrightarrow$ outputs

To learn the weights $\equiv$ function which the NN implements

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

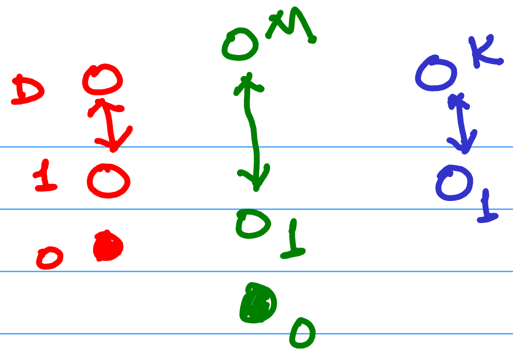
N points

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

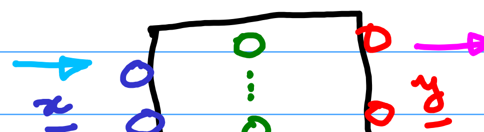
→ Output layer activation function
$$\sigma(a) = a, \quad y_k = a_k$$

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

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

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

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

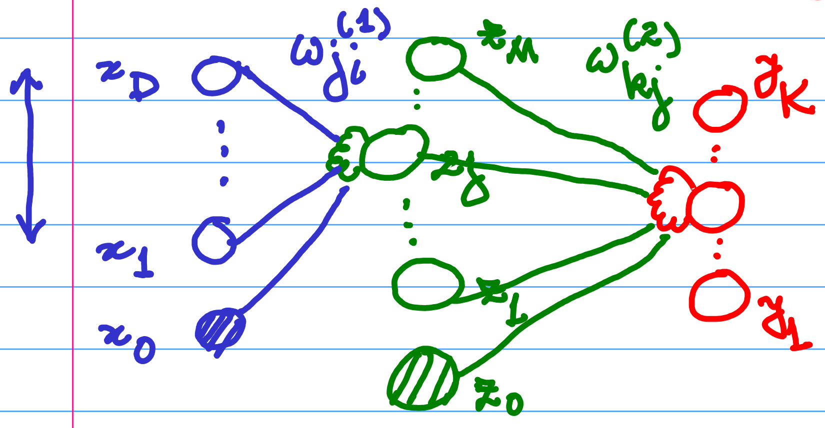
→ why only y_k? (defined only at the output layer)

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

t_k = k'th item of the target vector

A representative example

BACKPROPAGATION (contd.) [An example]



Assume

→ $h(\cdot)$: activation function is $\tanh()$

→ $\sigma(\cdot)$: activation function is a unit function

[2 layers of connections]

Why is this called 'Backpropagation'?

$\rightarrow \boxed{\underline{w}} \rightarrow$ Training phase : error is at the output : this is fed back to update the weights

$n \in \{1, N\}$

FOR each pattern in the training set in turn, we follow the following operations.

① A 'forward propagation'

(j: hidden layer)

$$a_j^{(1)} = \underline{w}_j^{(1)T} \underline{x} = \sum_{i=0}^{D} w_{ji}^{(1)} x_i$$

$$z_j = \tanh(a_j)$$

$$y_k = \underline{w}_k^{(2)T} \underline{z} = \sum_{j=0}^{M} w_{kj}^{(2)} z_j$$

② Evaluate δ_k's for each output unit

$$\delta_k = y_k - t_k$$

What is this, and how?

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

$$\delta_k \triangleq \frac{\partial E_n}{\partial a_k} = \frac{1}{2} \cdot 2\,(y_k - t_k)\frac{\partial y_k}{\partial a_k}$$

output layer activation

$\frac{\partial y_k}{\partial a_k} = 1$ as $y_k = a_k$ ($\sigma(\cdot)$ = unit fn.)

$$\delta_k = y_k - t_k$$ (Else, according to the specific activation function $\sigma(\cdot)$ at the output layer)

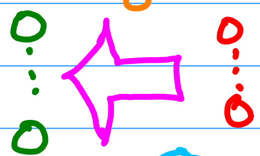
(3) **Backpropagate these to obtain δ_j's for the hidden layer units**

$$\delta_j = (1 - z_j^2)\sum_{k=1}^{K} w_{kj}\,\delta_k$$

What is this, and how?

δ_j → [hidden layer]

δ_k → previous step (step #2) [output layer]



$$\delta_j \triangleq \frac{\partial E_n}{\partial a_j} = \sum_{\forall k} \frac{\partial E_n}{\partial a_k}\,\frac{\partial a_k}{\partial a_j}$$

$\frac{\partial E_n}{\partial a_k} = \delta_k$ [step #2]

$$a_k = \underline{w}^{(2)T}\underline{z} = \sum_{j=0}^{M} w_{kj}^{(2)} z_j = \sum_{j=0}^{M} w_{kj}^{(2)} h(a_j)$$

$$\Rightarrow \frac{\partial a_k}{\partial a_j} = w_{kj}^{(2)}\frac{\partial h(a_j)}{\partial a_j} = w_{kj}^{(2)} h'(a_j)$$

$$= w_{kj}^{(2)}(1 - z_j^2)$$

$$\delta_j = \sum_{\forall k} \delta_k\, w_{kj}^{(2)}(1 - z_j^2) = (1 - z_j^2)\sum_{\forall k} w_{kj}^{(2)}\delta_k$$