CS 7015 Assignment - 2 Total sheet = 6

Name: Anoubhav Agarwaal
Roll no.: BE16B002

PAGE: 1
DATE: 1/1

Part - B

b Sigmoid ; $f(x) = 1/(1+e^{-x})$

By quotient rule,

$$f'(x) = \frac{(1+e^{-x})(0) - 1\times(-e^{-x})}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1+e^{-x}-1}{(1+e^{-x})^2} = \frac{1+e^{-x}}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2}$$

$$f'(x) = \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) = f(x)(1-f(x))$$

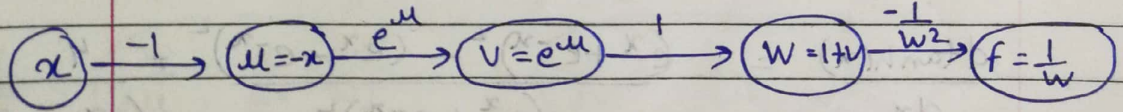
$\therefore$ Gradient of sigmoid $\sigma'(x) = \sigma(x)(1-\sigma(x))$

We can use a computation graph to represent a composite function as a network of nodes, where each node is a function.

Also, each edge represents the change in output node w.r.t input node i.e., the partial derivatives.

$\sigma(x) = \frac{1}{1+e^{-x}}$. Let $u = -x$, $v = e^u$, $w = 1+v$, $f = \frac{1}{w}$. Here, $u, v, w$ & $f$ are newly-defined intermediate variables.

The computational graph is given by,

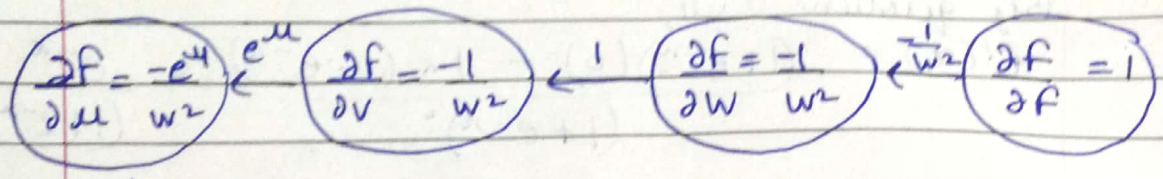



$$\frac{\partial u}{\partial x} = -1 \; ; \; \frac{\partial v}{\partial u} = e^u \; ; \; \frac{\partial w}{\partial v} = 1 \; ; \; \frac{\partial f}{\partial w} = -\frac{1}{w^2}$$

To obtain the gradient of the sigmoid, we perf

DATE : / . /

- orm reverse - mode differentiation ( backpropogation) on the computational graph.




$\downarrow -1$

$\frac{\partial f}{\partial x} = \frac{e^u}{w^2}$

$\therefore f(x) = \frac{1}{1+e^{-x}}$ ( on substituting w )

$\frac{\partial f}{\partial x} = \frac{e^u}{w^2}$

substituting u & w in terms of x

$\frac{\partial f}{\partial x} = \frac{e^{-x}}{(1+e^{-x})^2}$

This is same as the derivative obtained by using quotient rule.

And the decomposition used $\Rightarrow$ $\boxed{\sigma(x) = f(w(v(u(x))))}$

Chain rule : $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial f} \times \frac{\partial f}{\partial w} \times \frac{\partial w}{\partial v} \times \frac{\partial v}{\partial u} \times \frac{\partial u}{\partial x}$

2. Hyperbolic Tangent.

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

By quotient rule,

$\frac{d \tanh(x)}{dx} = \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2}$

$\llcorner (1*)$

$\frac{d \tanh(x)}{dx} = \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} = \frac{4}{(e^x + e^{-x})^2}$

$(a+b)^2 - (a-b)^2 = 4ab$ ; $a = e^x$ & $b = e^{-x}$

$\therefore \frac{d \tanh(x)}{dx} = \frac{4}{(e^x + e^{-x})^2}$

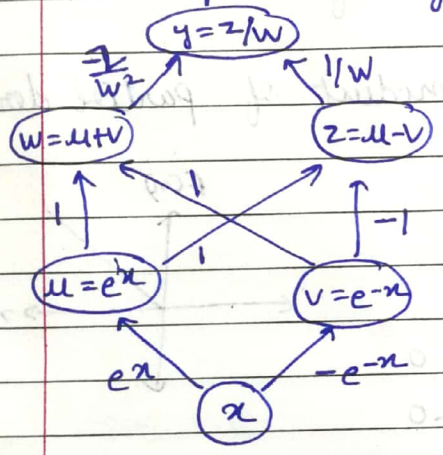
PAGE :
DATE : / /

In (1*), On splitting the fraction we obtain,

$$\frac{d\tanh(x)}{dx} = 1 - \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)^2 = 1 - \tanh^2(x)$$

Lets define new intermediate variables,
$u = e^x$, $v = e^{-x}$, $w = u+v$, $z = u-v$ and $y = \frac{z}{w}$.

The computational graph is given by,



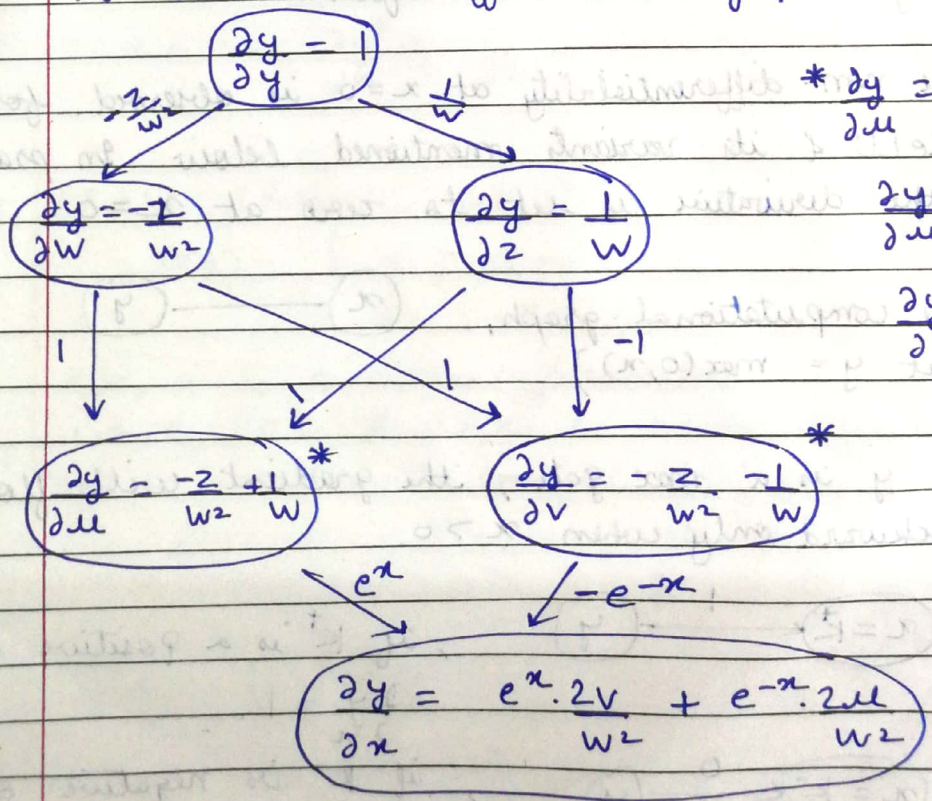

$$\frac{\partial u}{\partial x} = e^x \; ; \; \frac{\partial v}{\partial x} = -e^{-x}$$

$$\frac{\partial z}{\partial u} = 1 \; ; \; \frac{\partial w}{\partial u} = 1$$

$$\frac{\partial z}{\partial v} = 1 \; ; \; \frac{\partial z}{\partial v} = -1$$

$$\frac{\partial y}{\partial z} = \frac{1}{w} \; ; \; \frac{\partial y}{\partial w} = \frac{-z}{w^2}$$

The reverse-mode differentiation graph is given by,




$$* \frac{\partial y}{\partial u} = \frac{-z+w}{w^2}$$

$$\frac{\partial y}{\partial u} = \frac{2v}{w^2} \; ;$$

$$\frac{\partial y}{\partial v} = \frac{-2u}{w^2} \; ;$$

Substituting $u, v$ & $w$ in $\frac{\partial y}{\partial x}$,

PAGE :
DATE : / /

$$\frac{\partial y}{\partial x} = \frac{e^x \cdot 2e^{-x}}{(e^{-x}+e^x)^2} + \frac{e^{-x} \cdot 2e^x}{(e^{-x}+e^x)^2} = \frac{4}{(e^{-x}+e^x)^2}$$

This is same as the derivative obtained by using quotient rule.

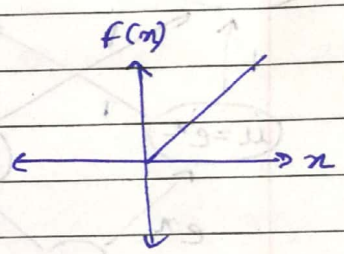
And the decomposition used ⇒

$$\tanh(x) = y(z(u(x), v(x)), w(u(x), v(x)))$$

The chain rule will involve the summation of all 'paths' (4 in this case) from node $x$ to node $y$.
Each path will be a product of partial derivatives

3. ReLU

$$f(x) = \max(0, x)$$

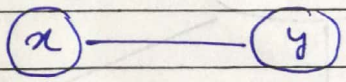



$$f'(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases}$$

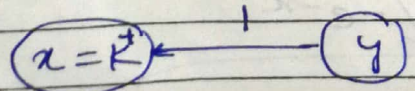
It is undefined at $x = 0$ as the left and right derivatives are not equal

The non-differentiability at $x = 0$ is observed for ReLU & its variants mentioned below. In practice, the derivative is set to zero at $x = 0$.

The computational graph, (Let $y = \max(0, x)$)

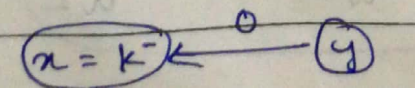



As $y$ is a max gate, the gradient will flow backward only when $x > 0$.




, if $k^+$ is a positive constant

$$\frac{\partial y}{\partial x} = 1.$$




, if $k^-$ is negative or zero

$$\partial y / \partial x = 0.$$

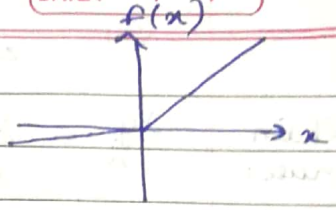
Anubhav Agarwal
BE16B002



PAGE :
DATE :

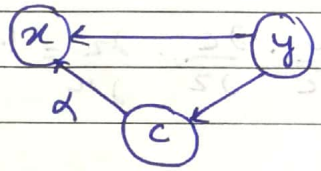
4. Leaky ReLU

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x & \text{otherwise} ; \alpha \geq 0 \end{cases}$$




Here, $\alpha$ is a positive constant set by the user.

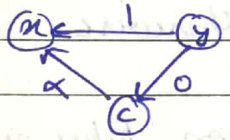
The computational graph is given by:
(Let $c = \alpha x$ be an intermediate variable)



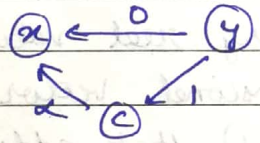

$y = f(x)$

$\frac{\partial c}{\partial x} = \alpha$

If $x > 0$,




If $x < 0$,




$$\therefore f'(x) = \begin{cases} 1, & x > 0 \\ \alpha & \text{otherwise} \end{cases}$$

Chain rule:

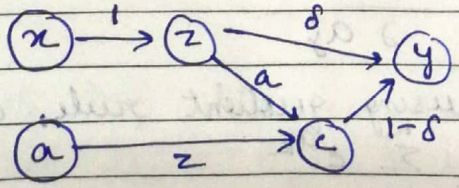
(i) $x > 0$ $\quad \frac{\partial y^*}{\partial x} = \frac{\partial y}{\partial x} + \frac{\partial y}{\partial c} \cdot \frac{\partial c}{\partial x}^{0} = 1$

(ii) $x < 0$ $\quad \frac{\partial y^*}{\partial x} = \frac{\partial y}{\partial x}^{0} + \frac{\partial y}{\partial c} \cdot \frac{\partial c}{\partial x} = 1 \cdot \alpha = \alpha$

5. PReLU (Parametric ReLUs)

$$f(x) = \begin{cases} x, & x > 0 \\ ax & \text{otherwise} \end{cases}$$

Here, a is a learned parameter.
Let $c = az$ & $z = x$ be newly defined interm-ediate variables. The computational graph is given by:

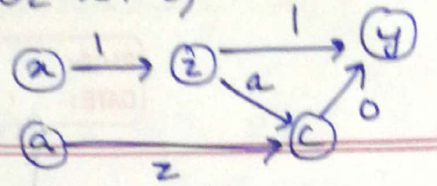



$y = f(x)$

$\frac{\partial z}{\partial x} = 1$ ; $\frac{\partial c}{\partial a} = z$ ; $\frac{\partial c}{\partial z} = a$

if $x > 0$, $\delta = 1$
else $\delta = 0$

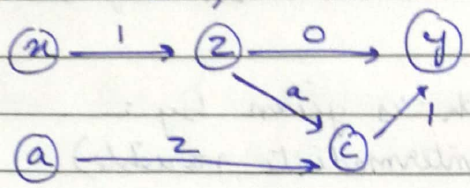
For $x > 0$,




PAGE :
DATE : / /

Chain rule: $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial x} + \frac{\partial y}{\partial c} \cdot \frac{\partial c}{\partial z} \cdot \frac{\partial z}{\partial x} = 1 \cdot 1 = 1$ //

For $x \leq 0$,




Function decomposition
$f(x) = y(z(x), c(z(x), a))$

$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial x} + \frac{\partial y}{\partial c} \cdot \frac{\partial c}{\partial z} \cdot \frac{\partial z}{\partial x} = 1 \cdot a \cdot 1 = a$ //

$$f'(x) = \begin{cases} 1, & x > 0 \\ a & \text{otherwise} \end{cases}$$

7. Softmax : The function takes an N-dimensional vector of arbitrary real values and produces another N-dimensional vector with real values in the range (0, 1) that adds up to 1.

Notation :

$$S(a) = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \rightarrow \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_N \end{bmatrix}$$

$S(a) : \mathbb{R}^N \rightarrow \mathbb{R}^N$

∴ N gradients w.r.t N variables need to be calculated i.e., N×N total gradients

where $S_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \quad \forall j \in 1..N$

The partial derivative of the $i^{th}$ output ($S_i$) w.r.t the $j^{th}$ ($a_j$) input is given by $D_j S_i$.

$$D_j S_i = \frac{\partial S_i}{\partial a_j} = \frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j}$$

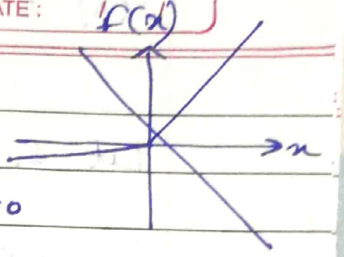
This is $\frac{g(x)}{h(x)}$ form

We will solve this using quotient rule, where $g_i = e^{a_i}$ & $h_i = \sum_{k=1}^{N} e^{a_k}$

PAGE :
DATE :

4. Leaky ReLU

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x & \text{otherwise} \end{cases}, \quad \alpha \geq 0$$




$$h_i = \sum_{k=1}^{N} e^{a_k}$$

$$\frac{dh_i}{da_j} = e^{a_j} \quad \forall\, j \in 1 \ldots N$$

$$\frac{dg_i}{da_j} = \begin{cases} e^{a_j} & \forall\, i = j, \text{ otherwise } 0 \\ 0 & \text{otherwise} \end{cases}$$

① For $i = j$ case, the quotient rule gives us:

$$D_j S_i \underset{(i=j)}{=} \frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j} = \frac{e^{a_i} \sum_{k=1}^{N} e^{a_k} - e^{a_j} e^{a_i}}{\left(\sum_{k=1}^{N} e^{a_k}\right)^2}$$

$$D_j S_i \underset{(i=j)}{=} \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}} - \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \cdot \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}} = S_i - S_j S_i$$

$$\therefore D_j S_i \underset{(i=j)}{=} S_i (1 - S_j) \quad \text{— (1)}$$

② For $i \neq j$ case, the quotient rule gives us:

$$D_j S_i \underset{i \neq j}{=} \frac{\sum_{k=1}^{N} e^{a_k} \cdot 0 - e^{a_j} e^{a_i}}{\left(\sum_{k=1}^{N} e^{a_k}\right)^2} = \frac{-e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \cdot \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}$$

$$D_j S_i \underset{(i \neq j)}{=} - S_i S_j \quad \text{— (2)}$$

From (1) & (2),

$$\boxed{D_j S_i = \begin{cases} S_i (1 - S_j), & i = j \\ -S_i S_j & i \neq j \end{cases}}$$

Lets try obtaining the computational graph,

Let $u_i = e^{a_i} \quad \forall\, i = 1 \ldots N$ be intermediate variables

$$t = \sum_{i=1}^{N} u_i$$

Remember the softmax function is the following

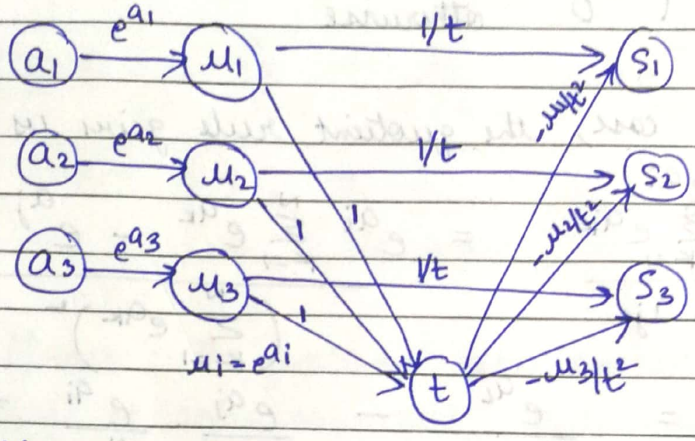
PAGE:
DATE: / /

$$S(a): \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \rightarrow \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix}$$

Let $s_i = \frac{u_i}{t} \quad \forall i = 1..N$ and $a_i$ are the input variables.

We define $t = \sum_{i=1}^{N} u_i$ to reduce the number of paths in the graph.

For $N = 3$, (can be easily extended for large N), the graph is given by:




$$\frac{\partial t}{\partial u_i} = \frac{\partial \sum u_k}{\partial u_i} = 1$$

$$\frac{\partial s_i}{\partial u_i} = \frac{1}{t}$$

$$\frac{\partial s_i}{\partial t} = \frac{\partial \left(\frac{u_i}{t}\right)}{\partial t} = -\frac{u_i}{t^2}$$

If we densely connected all $u_i$'s with $s_i$'s we would have $n \times n$ connections. By introducing $t = \sum u_i$, we have $n$ (connections from $u_i$ to $t$) + $n$ ($u_i$ to $s_i$) + $n$ ($t$ to $s_i$) = $3n$.

For $N = 3$, $n^2 == 3n$ will have same no. of connections. But for large N, this won't be the case. Also, the partial derivatives in the graph are simplified after introducing intermediate $t$.

$$\frac{\partial s_1}{\partial a_1} = \sum_{\text{all paths from } a_1 \text{ to } s_1} \text{Product of the PDs. in path}$$

$$\frac{\partial S_1}{\partial a_1} = \frac{\partial S_1}{\partial u_1} \cdot \frac{\partial u_1}{\partial a_1} + \frac{\partial S_1}{\partial t} \cdot \frac{\partial t}{\partial u_1} \cdot \frac{\partial u_1}{\partial a_1}$$

$$\frac{\partial S_1}{\partial a_1} = \frac{1}{t} \cdot e^{a_1} + \left(-\frac{u_1}{t^2}\right) \cdot 1 \cdot e^{a_1}$$

$$\frac{\partial S_1}{\partial a_1} = \frac{e^{a_1}}{t}\left(1 - \frac{u_1}{t}\right) = \frac{u_1}{t}\left(1 - \frac{u_1}{t}\right) = S_1(1-S_1)$$

$$\therefore \boxed{\frac{\partial S_1}{\partial a_1} = S_1(1-S_1)}$$ follows the $D_j S_i = S_i(1-S_i)$ $(i=j)$

$$\frac{\partial S_3}{\partial a_2} = \sum_{\text{all paths from } a_2 \text{ to } S_3} \text{Product of PDs in the path}$$

$$\frac{\partial S_3}{\partial a_2} = \frac{\partial S_3}{\partial t} \cdot \frac{\partial t}{\partial u_2} \cdot \frac{\partial u_2}{\partial a_2} = -\frac{u_3}{t^2} \cdot 1 \cdot e^{a_2}$$

$$\frac{\partial S_3}{\partial a_2} = -\frac{u_3 \cdot u_2}{t^2} = -S_3 S_2$$

$$\therefore \boxed{\frac{\partial S_3}{\partial a_2} = -S_3 S_2}$$ follows the $D_j S_i = -S_j S_i$ $(i \neq j)$

∴ The function decomposition of

$y_i = S_i \Rightarrow y_i = S_i\left(u_i(a_i), t(u_1, .., u_N)\right)$

×

Q.2 solved later

Q3. Hand-calculation of gradients

Notation (From Nielsen's book)

1) We'll use $w^l_{jk}$ to denote the weight for the connection from the $K^{th}$ neuron in the $(l-1)^{th}$ layer to the $j^{th}$ neuron in the $l^{th}$ layer.
e.g. $w^3_{12} = 0.1$, $w^2_{21} = -0.2$

2) $b^l_j$ for the bias of the $j^{th}$ neuron in the $l^{th}$ layer. $a^l_j$ for the activation of the $j^{th}$ neuron in the $l^{th}$ layer.

3) we define an intermediate quantity, $z_j^l$, this is the weighted input to the $j^{th}$ neuron in layer $l$.

$$z_j^l = \sum_k w_{jk}^l \times a_k^{l-1} + b_j^l$$
$$a_j^l = \sigma(z_j^l)$$

4) $w^l$ is the weight matrix of a layer, the entries of which are $w_{jk}^l$. similarly, $b^l$, $a^l$, $z^l$, $\delta^l$.

5) we define the error $\delta_j^l$ of neuron $j$ in layer $l$ by $\delta_j^l = \frac{\partial C}{\partial z_j^l}$, where $C$ is the cost function

6) The equations being used.

i) $\boxed{\delta^L = \frac{\partial C}{\partial a^L} \odot \sigma'(z^L)}$ ; Error in last layer.

$L$: no. of layers, $\odot$ is the hadamard product

ii) $\boxed{\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)}$ ; Error in layer $l$ given the error in layer $l+1$.

iii) $\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \times \frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \times \frac{\partial(\sum_k w_{jk}^l a_k^{l-1} + b_j^l)}{\partial b_j^l}$

$\boxed{\frac{\partial C}{\partial b_j^l} = \delta_j^l \times 1 = \delta_j^l}$ ; Gradient of cost function w.r.t $b_j^l$ is just $\delta_j^l$.

iv) $\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \times \frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l \times \frac{\partial(\sum_k w_{jk}^l a_k^{l-1} + b_j^l)}{\partial w_{jk}^l}$

$\boxed{\frac{\partial C}{\partial w_{jk}^l} = \delta_j^l \times a_k^{l-1}}$

. Basic steps in the algorithm:

1) Feedforward
2) Calculate output error
3) Backpropogate the error
4) Gradient descent
5) Repeat

Step 1: Feedforward

$z_1^2 = 0.2 \times 0.8 + 0.3 \times 0.7 + 0.4 \times 0.1 - 0.2$

$z_1^2 = 0.210$

$a_1^2 = \sigma(z_1^2) = 0.552$

$z_2^2 = -0.2 \times 0.8 + 0.1 \times 0.7 + 0.5 \times 0.1 - 0.2$

$z_2^2 = -0.240$

$a_2^2 = \sigma(z_2^2) = 0.440$

$z_1^3 = 0.552 \times -0.3 + 0.440 \times 0.1 - 0.2$

$z_1^3 = -0.322$

$a_1^3 = \sigma(z_1^3) = 0.420$

Step 2: Calculate output error

Loss function: $L(a^L) = (y - a^L)^2$

$L = (y - a_1^3)^2 = (0.9 - 0.42)^2 = 0.2304$

Step 3: Backpropogate the error

(Eqn(i) in Notation)

$$\frac{\partial L}{\partial z_1^3} = \delta_1^3 = \frac{\partial L}{\partial a_1^3} \times \frac{\partial a_1^3}{\partial z_1^3} = \frac{\partial L}{\partial a_1^3} \times \sigma'(z_1^3)$$

$$\delta_1^3 = \frac{\partial (y - a_1^3)^2}{\partial a_1^3} \times \sigma(z_1^3)(1 - \sigma(z_1^3))$$

$\delta_1^3 = (a_1^3 - y) \times \sigma(-0.322)(1 - \sigma(-0.322))$

$\delta_1^3 = -0.117$ (substitute $a_1^3 = 0.42$, $y = 0.9$ & $\sigma(-0.322) = 0.42$)

From eqn (iii) in Notation,

$$\frac{\partial L}{\partial b_1^3} = \delta_1^3 = -0.117$$

From eqn (iv) in Notation,

$$\frac{\partial L}{\partial w_{11}^3} = \delta_1^3 \times a_1^2 = -0.117 \times 0.552 = -0.065$$

$$\frac{\partial L}{\partial w_{12}^3} = \delta_1^3 \times a_2^2 = -0.117 \times 0.440 = -0.051$$

From eqn (ii) in Notation,

$$\frac{\partial L}{\partial z^2} = \delta^2 = (w^3)^T \delta^3 \odot \sigma'(z^2)$$

$$\delta^2 = \begin{bmatrix} \delta_1^2 \\ \delta_2^2 \end{bmatrix} = [w_{11}^3 \; w_{12}^3]^T [\delta_1^3] \odot \begin{bmatrix} \sigma'(z_1^2) \\ \sigma'(z_2^2) \end{bmatrix}$$

$$\delta^2 = \begin{bmatrix}\delta_1^2\\ \delta_2^2\end{bmatrix} = \begin{bmatrix}-0.3\\ 0.1\end{bmatrix}\begin{bmatrix}-0.117\end{bmatrix} \odot \begin{bmatrix}\sigma'(0.210)\\ \sigma'(-0.240)\end{bmatrix}$$

PAGE:
DATE:

$$\delta^2 = \begin{bmatrix}0.035\\ -0.012\end{bmatrix} \odot \begin{bmatrix}0.247\\ 0.246\end{bmatrix} = \begin{bmatrix}0.009\\ -0.003\end{bmatrix}$$

$$\frac{\partial L}{\partial b_1^2} = \delta_1^2 = \underline{0.009} \qquad \frac{\partial L}{\partial b_2^2} = \delta_2^2 = \underline{-0.003}$$

$$\frac{\partial L}{\partial w_{jk}^2} = \delta_j^2 \times a_k^1 \quad ; j = 1,2 \quad k = 1,2,3$$

$$\frac{\partial L}{\partial w^2} = \begin{bmatrix}\delta_1^2 \cdot a_1^1 & \delta_1^2 \cdot a_2^1 & \delta_1^2 \cdot a_3^1\\ \delta_2^2 \cdot a_1^1 & \delta_2^2 \cdot a_2^1 & \delta_2^2 \cdot a_3^1\end{bmatrix}_{2\times 3}$$

$$\frac{\partial L}{\partial w^2} = \begin{bmatrix}0.009\times 0.8 & 0.009\times 0.7 & 0.009\times 0.1\\ -0.003\times 0.8 & -0.003\times 0.7 & -0.003\times 0.1\end{bmatrix}$$

$$\frac{\partial L}{\partial w^2} = \begin{bmatrix}7.2\times 10^{-3} & 6.3\times 10^{-3} & 9\times 10^{-4}\\ -2.4\times 10^{-3} & -2.1\times 10^{-3} & -3\times 10^{-4}\end{bmatrix}$$

**Step 4: Gradient descent**

$$w_{jk}^l \rightarrow w_{jk}^l - \eta\, \partial L / \partial w_{jk}^l$$
$$b_j^l \rightarrow b_j^l - \eta\, \partial L / \partial b_j^l$$

For $\eta = 1$,

$w_{11}^2 = 0.2 - 1\times 7.2\times 10^{-3} = 0.1928$ $\qquad b_1^2 = -0.2 - 0.009$

$w_{12}^2 = 0.3 - 1\times 6.3\times 10^{-3} = 0.2937$ $\qquad b_1^2 = \underline{-0.209}$

$w_{13}^2 = 0.4 - 9\times 10^{-4} = 0.3991$ $\qquad b_2^2 = -0.2 + 0.003$

$b_2^2 = \underline{-0.197}$

$w_{21}^2 = -0.2 + 1\times 2.4\times 10^{-3} = -0.1976$

$w_{22}^2 = 0.1 + 1\times 2.1\times 10^{-3} = 0.1021$

$w_{23}^2 = 0.5 + 1\times 3\times 10^{-4} = 0.5003$

$w_{11}^3 = -0.3 + 1\times 0.065 = -0.235$

$w_{12}^3 = 0.1 + 1\times 0.051 = 0.151$

**Step 5: Repeat**

4

EE6002

Moulshav

Forward pass with updated weights :

$z_1^2 = 0.1928 \times 0.8 + 0.2937 \times 0.7 + 0.3991 \times 0.1 - 0.209$

PAGE:
DATE: / /

$z_1^2 = \underline{0.1907}$

$a_1^2 = \sigma(z_1^2) = \underline{0.5475}$

$z_2^2 = -0.1976 \times 0.8 + 0.1024 \times 0.7 + 05003 \times 0.1 - 0.197$

$z_2^2 = \underline{-0.2536}$

$a_2^2 = \underline{0.4419} \quad \sigma(z_2^2)$

$z_1^3 = 0.5475 \times -0.235 + 0.4419 \times 0.151 - 0.083$

$z_1^3 = \underline{-0.1449}$

$a_1^3 = \sigma(z_1^3) = \underline{0.4638}$

$Loss = (0.9 - 0.4638)^2 = 0.1903$

∴ After backpropogating +G.D after 1 training example update, we get

| | Original | After 1 update | |
|---|---|---|---|
| $\hat{y}$ | 0.42 | 0.4638 | Target = 0.9 |
| L | 0.2304 | 0.1903 | |

---

2. Gradient calculation of common loss functions

$y_i$ : true output of neuron i, $a_i^L$ : final activation of neuron i

1) Cross entropy loss

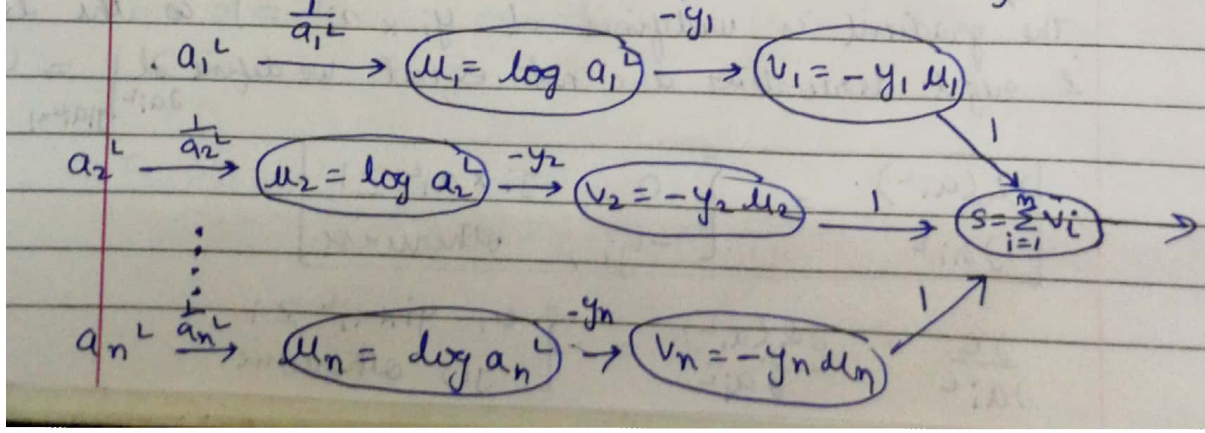
$$C = -\sum_{i=1}^{n} y_i \log(a_i^L)$$

n is the no. of neuron in final activation layer.

L is the last layer.

$$\boxed{\frac{\partial C}{\partial a_i^L} = -\frac{y_i}{a_i^L}}$$

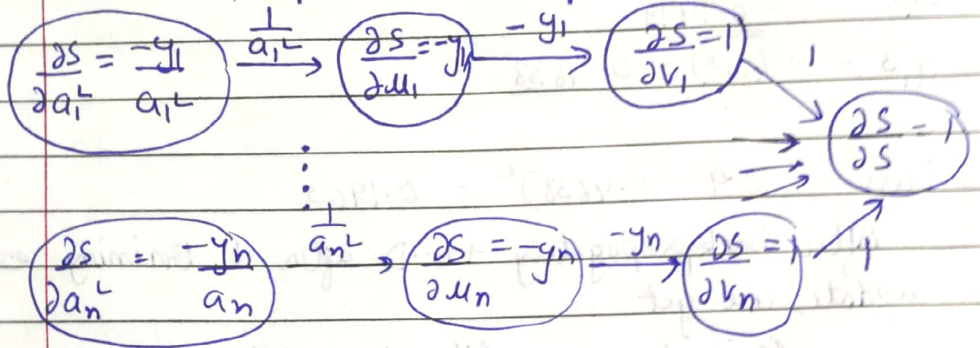
The computational graph of C is given by,



Let $u_i = \log a_i^L$ & $v_i = -y_i u_i$ and $S = \sum_{i=1}^{n} v_i$ be intermediate variables.

The edges of the graph represent the change in output node w.r.t input node, i.e, partial derivative. $\frac{\partial u_i}{\partial a_i^L} = \frac{1}{a_i^L}$, $\frac{\partial v_i}{\partial u_i} = -y_i$, $\frac{\partial S}{\partial v_i} = 1$

To obtain the gradient of the cross-entropy loss with respect to output activations, we perform reverse-mode differentiation (backpropogation) on the computational graph.




$\frac{\partial C}{\partial a_i^L} = \frac{\partial S}{\partial a_i^L} \cdot \frac{\partial S}{\partial a_i^L} = \frac{-y_i}{a_i^L}$, this is same as what was obtained by differentiation earlier.

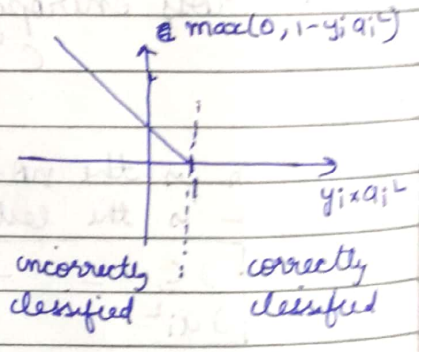
Decomposition of function used $\Rightarrow$

$C(a_1^L, .., a_n^L) = S(v_1(u_1(a_1^L)), ...., v_n(u_n(a_n^L)))$

2) Hinge loss

$C(a_1^L, .., a_n^L) = \sum_{i=1}^{n} \max(0, 1 - y_i \times a_i^L)$

$= \sum_{i=1}^{n} l_{y_i}(a_i^L)$




For a single neuron.

$l_{y_i}(a_i^L) = \max(0, 1 - y_i \times a_i^L)$

The gradient is undefined at $y_i \times a_i^L = 1$ as the left & right derivatives are not equal. We define $\frac{\partial l}{\partial a_i^L}\Big|_{y_i a_i^L = 1} = 0$.

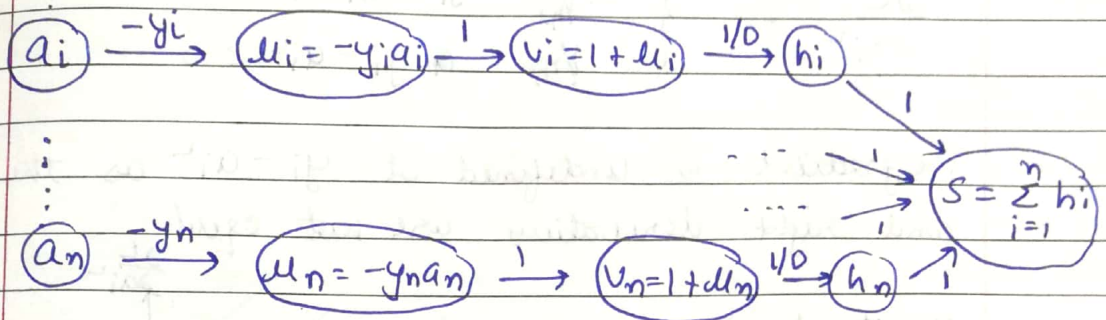
$$\boxed{\frac{\partial l_{y_i}(a_i^L)}{\partial a_i^L} = \begin{cases} 0, & y_i \times a_i^L \geq 1 \\ -y_i, & \text{otherwise} \end{cases}}$$

$$\frac{\partial C}{\partial a_i^L} = \frac{\partial l_{y_i}(a_i^L)}{\partial a_i^L} = \begin{cases} 0, & y_i \times a_i^L \geq 1 \\ -y_i, & \text{otherwise} \end{cases}$$

Let $u_i = -y_i a_i$, $v_i = 1 + u_i$, $h_i = \max(0, v_i)$ and $S = \sum h_i$ be intermediate variables.

DATE: / /

The computational graph is given by,

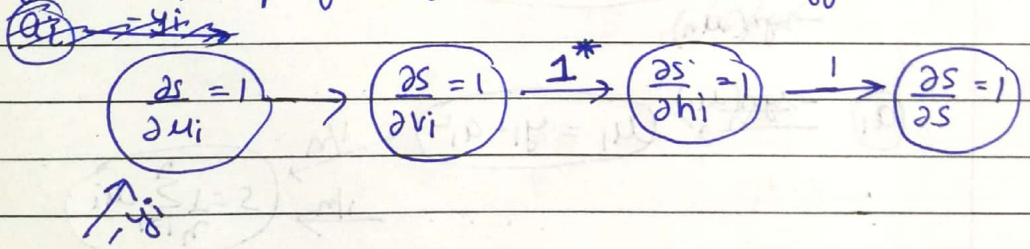



$$\frac{\partial u_i}{\partial a_i} = -y_i \; ; \; \frac{\partial v_i}{\partial u_i} = 1, \; \frac{\partial S}{\partial h_i} = 1$$

$h_i$ is a max gate. The gradient will be 1 if $v_i > 0$ and 0 if $v_i < 0$

∴ If $a_i$ (the predicted label) is same as the target $y_i$, i.e., it is correctly classified, ~~with high confidence~~ then the gradient from $v_i \xrightarrow{0} h_i$ will be zero.

If $v_i > 0$, performing reverse-mode differentiation,




$$\frac{\partial S}{\partial a_i} = -y_i$$

∴ $v_i > 0$, i.e., $1 - y_i a_i > 0 \Rightarrow y_i a_i < 1$

we get $\frac{\partial C}{\partial a_i} = -y_i$

(* becomes 0)

When $v_i \leq 0$, i.e., $1 - y_i a_i \leq 0 \Rightarrow y_i a_i \geq 1$

we get $\frac{\partial C}{\partial a_i} = 0$

We obtain same results as differentiation which is

$$\frac{\partial C}{\partial a_i} = \begin{cases} 0, & y_i a_i \geq 1 \\ -y_i & \text{otherwise.} \end{cases}$$

Functional decomposition:

$$C(a_1^L, \ldots, a_n^L) = S(h_1(v_1(u_1(a_1^L))), \ldots, h_n(v_n(u_n(a_n^L))))$$

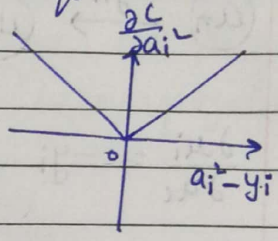
3) L1 loss

$$C = \frac{1}{n}\sum_{i=1}^{n} |y_i - a_i^L|$$

PAGE:
DATE: / /

$$\frac{\partial C}{\partial a_i^L} = \begin{cases} -1/n, & y_i - a_i^L > 0 \\ 1/n, & y_i - a_i^L < 0 \end{cases}$$

The gradient is undefined at $y_i = a_i^L$ as the left and right derivatives are not equal.

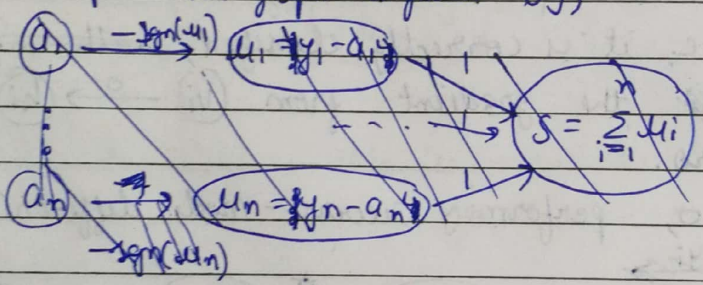


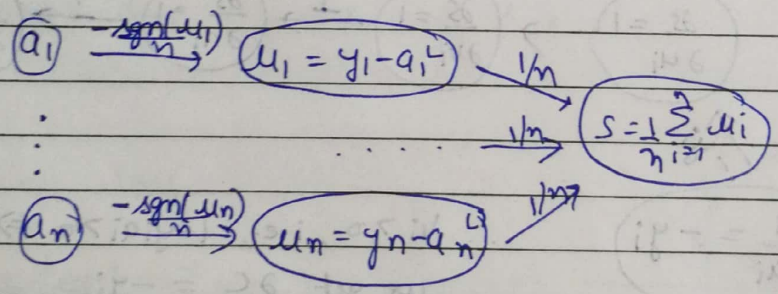

Shorthand: $\frac{\partial C}{\partial a_i^L} = \frac{-1}{n}\,\text{sgn}(y_i - a_i^L)$

$\text{sgn}(x)$ = signum function $= \begin{cases} 0, & x = 0 \\ 1, & x > 0 \\ -1, & x < 0 \end{cases}$

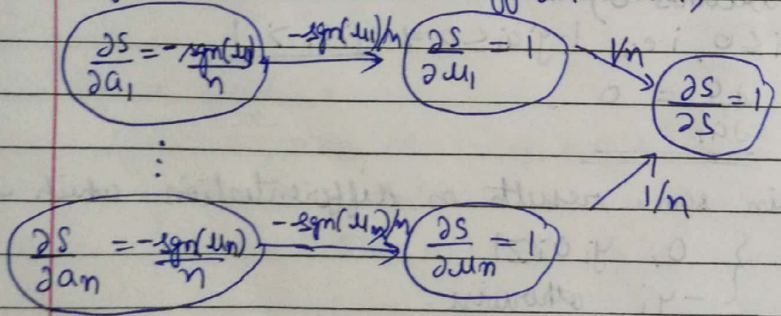
Let $u_i = y_i - a_i^L$ and $S = \frac{\sum_{i=1}^{n} u_i}{n}$ be intermediate variables.

The computational graph is given by,




After reverse-mode differentiation,




$$\therefore \frac{\partial S}{\partial a_i} = \frac{-\text{sgn}(u_i)}{n} = \frac{-\text{sgn}(y_i - a_i^L)}{n}$$

We obtain the same result as differentiation.

5 | The functional decomposition is given by,

$C(a_i^L, \ldots, a_n^L) = S(u_1(a_i^L), \ldots, u_n(a_n^L))$

PAGE :
DATE : / /

Anubhav
3E16 Book2

4) Huber loss

(Wikipedia) → H

$$L_{\delta}(y, f(x)) = \begin{cases} 0.5\,(y - f(x))^2, & |y - f(x)| \le \delta \\ \delta\,|y - f(x)| - \frac{\delta^2}{2} & \text{otherwise} \end{cases}$$

For our notation, $f(x) = a_i^L$. Lets define $R_i$ as the residual. $R_i = y_i - f(x_i) = y_i - a_i^L$

$$\underset{\delta, y_i}{\ell}(R_i) = \begin{cases} 0.5\,R_i^2, & |R_i| \le \delta \\ \delta\,|R_i| - \frac{\delta^2}{2}, & |R_i| > \delta \end{cases} \quad ; \quad \underbrace{C = \frac{1}{n}\sum_{i=1}^{n} \ell_{\delta, y_i}(R_i)}_{\text{cost function}}$$

For small residual $R_i$, the huber function $\ell(R_i)$ reduces to the L2 loss, and for large $R_i$ it reduces to the robust L1 loss. The derivatives are continuous at $|R_i| = h$.

$$\frac{d\ell}{dR_i} = \begin{cases} R_i, & |R_i| \le \delta \\ \delta\,\text{sign}(R_i), & |R_i| > \delta \end{cases} = \begin{cases} -\delta, & R_i < -\delta \\ R_i, & |R_i| \le \delta \\ \delta, & R_i > \delta \end{cases}$$

$\left(\frac{dC}{da_i^L} = \frac{dC}{d\ell} \times \frac{d\ell}{a_i^L} = \eta \cdot \frac{d\ell}{da_i^L}\right)$

$\frac{d\ell}{dR_i}$

$$\frac{dC}{da_i^L} = \frac{dC}{dR_i} \times \frac{dR_i}{\partial a_i^L} = -\frac{dC}{R_i} = \begin{cases} \delta, & y_i - a_i^L < -\delta \\ a_i^L - y_i, & |y_i - a_i^L| \le \delta \\ -\delta, & y_i - a_i^L > \delta \end{cases}$$

$$\boxed{\frac{dC}{da_i^L} \cdot \eta = \begin{cases} \delta, & y_i - a_i^L < -\delta \\ a_i^L - y_i, & |y_i - a_i^L| \le \delta \\ -\delta, & y_i - a_i^L > \delta \end{cases}}$$

Lets try to decompose this function C using a computational graph. $C = \frac{1}{n}\sum_{i=1}^{n} \ell_{\delta, y_i}(R_i)$.

For simplicity, lets first find the computational graph of $\ell_{\delta, y_i} = \begin{cases} 0.5\,R_i^2, & |R_i| \le \delta \\ \delta\,|R_i| - \frac{\delta^2}{2}, & |R_i| > \delta \end{cases}$

Let $t = y_i - a_i$ ; $u = \frac{t^2}{2}$ ; $v = \delta|t|$ ;

$w = v - \frac{\delta^2}{2}$ ; $x = (|t| \le \delta)\ ?\ u : 0$ ;

$y = (|t| > \delta)\ ?\ w : 0$ ; be intermediate variables.

We see that $t$ is now same as the residual $R_i$.

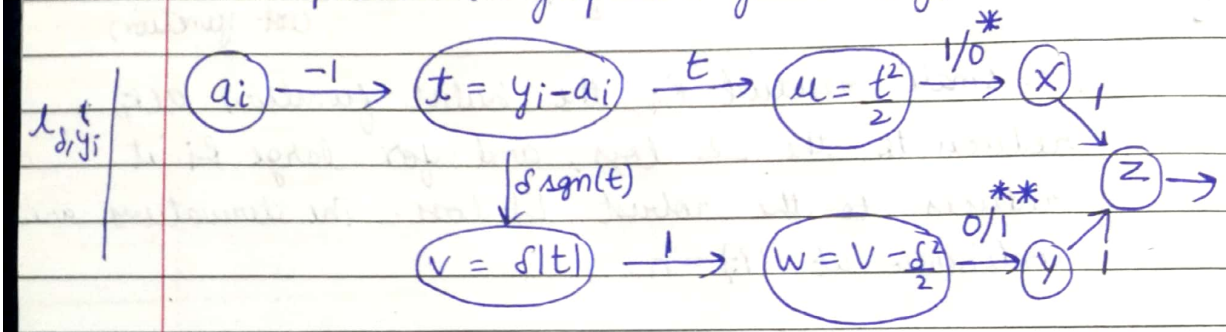
x & Y are ternary operators

condition ? value_if_true : value_if_false.

This is required as $l_{\delta, y_i}(R_i)$ has different value when $|R_i| \le \delta$ and when $|R_i| > \delta$

Also, define $z = x + y$

The computational graph is given by,




$\frac{\partial t}{\partial a_i} = -1$ ; $\frac{\partial u}{\partial t} = t$ ; $\frac{\partial v}{\partial t} = \frac{\partial \delta|t|}{\partial t} = \delta\, sgn(t)$

$\frac{\partial w}{\partial v} = 1$ ; $\frac{\partial z}{\partial x} = 1$ ; $\frac{\partial z}{\partial y} = 1$ ;

$x = (|t| \le \delta)\ ?\ u : 0$

$y = (|t| > \delta)\ ?\ w : 0$

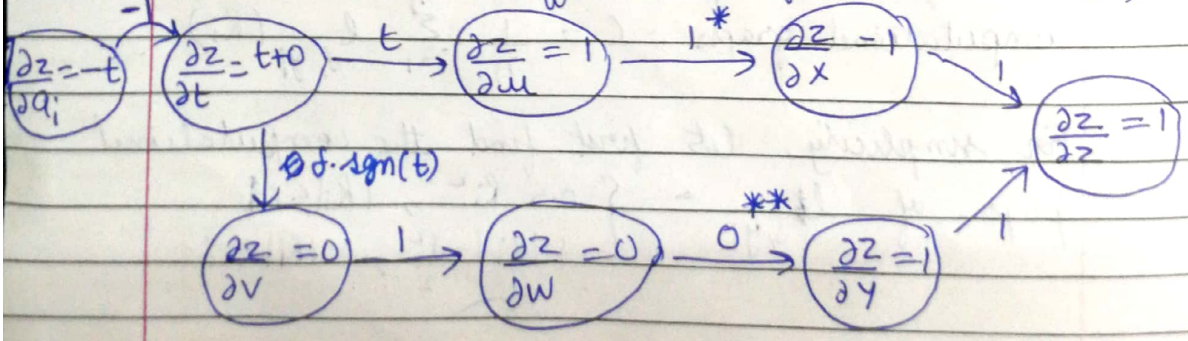
Case 1: If $|t| = |R_i| \le \delta$ then, $x = u$ and $y = 0$

Case 2: If $|t| = |R_i| > \delta$ then, $x = 0$ and $y = w$

$(|R_i| \le \delta)$ In case 1: the edge pd at * will be $\frac{\partial x}{\partial u} = \underline{1}$ &

" " " " ** will be $\frac{\partial y}{\partial w} = \underline{0}$

The reverse-mode differentiation for case 1 ($|R_i| \le \delta$)



We get $\frac{\partial z}{\partial a_i} = -t = -(y_i - a_i) = a_i - y_i$

PAGE :
DATE : / /

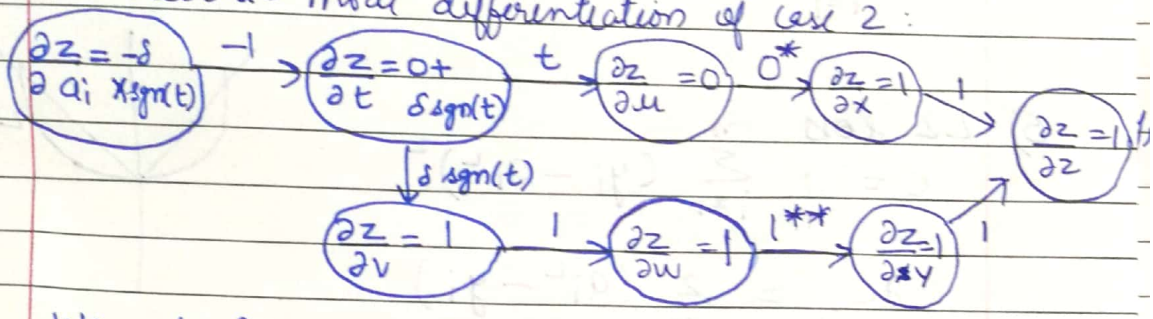
(1)

i.e., when $|R_i| \leq \delta \Rightarrow \partial z / \partial a_i = a_i - y_i$

In case 2, $|R_i| > \delta$ i.e., $R_i \geq \delta$ or $R_i \leq -\delta$

$\frac{\partial x}{\partial u} = 0$ & $\frac{\partial y}{\partial w} = 1$

The reverse-mode differentiation of case 2:




We get $\frac{\partial z}{\partial a_i} = -\delta \, sgn(t) = \begin{cases} -\delta, & t > 0 \\ \delta, & t < 0 \end{cases}$

$t == R_i \Rightarrow (y_i - a_i) \Rightarrow \begin{cases} -\delta, & R_i > 0 \\ \delta, & R_i < 0 \end{cases}$

But, we know that for case 2, $R_i > \delta$ or $R_i < -\delta$

$\therefore \frac{\partial z}{\partial a_i} = \begin{cases} -\delta, & R_i > \delta \\ \delta, & R_i < -\delta \end{cases}$ — (2)

Combining (1) from case i & (2) from case ii,

$$\frac{\partial z}{\partial a_i} = \frac{\partial l_{\delta, y_i}}{\partial a_i} = \begin{cases} \delta, & R_i < -\delta \\ a_i - y_i, & |R_i| \leq \delta \\ -\delta, & R_i > \delta \end{cases}$$

$$\frac{\partial l_{\delta, y_i}}{\partial a_i} = \begin{cases} \delta, & y_i - a_i < -\delta \\ a_i - y_i, & |y_i - a_i| \leq \delta \\ -\delta, & y_i - a_i > \delta \end{cases}$$

$$\frac{\partial C}{\partial a_i} = \frac{\partial C}{\partial l_{\delta, y_i}} \cdot \frac{\partial l_{\delta, y_i}}{\partial a_i} = \frac{\partial \left( \sum_{i=1}^{n} \frac{1}{n} \partial l_{\delta, y_i} \right)}{\partial l_{\delta, y_i}} \cdot \frac{\partial l_{\delta, y_i}}{\partial a_i}$$

$$\frac{\partial C}{\partial a_i} = \frac{1}{n} \cdot 1 \cdot \frac{\partial l_{\delta, y_i}}{\partial a_i} = \frac{1}{n} \frac{\partial l_{\delta, y_i}}{\partial a_i}$$

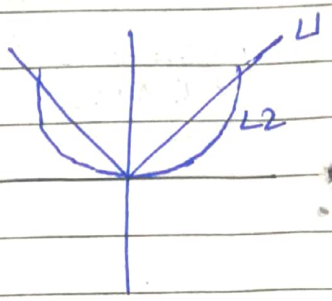
$$\frac{\partial C}{\partial a_i} \cdot n = \begin{cases} \delta, & y_i - a_i^L < -\delta \\ a_i^L - y_i, & |y_i - a_i^L| \le \delta \\ -\delta, & y_i - a_i^L > \delta \end{cases}$$

PAGE:
DATE: / /

This is same as what we obtain using normal differentiation.

The functional decomposition of the Huber loss:

$$C(a_1^L, \ldots, a_n^L) = \sum_{i=1}^{n} z_i \left( x_i(u_i(t_i(a_i^L))) + y_i(w_i(v_i(t_i(a_i^L)))) \right)$$




5) L2 loss

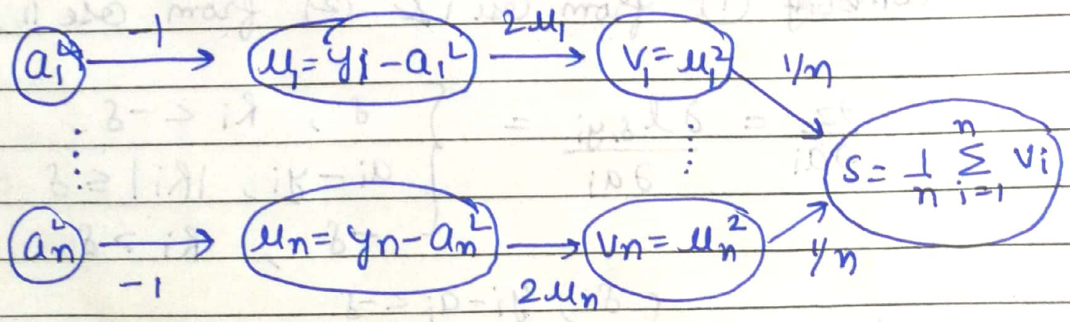
$$C = \frac{1}{n} \sum_{i=1}^{n} (y_i - a_i^L)^2$$

$$\frac{\partial C}{\partial a_i^L} = \frac{2}{n} (a_i^L - y_i)$$

The L1 loss is constant irrespective of whether $a_i^L$ is close to / far from $y_i$. Whereas, L2 loss decreases more rapidly than L1 when $a_i^L$ is far from $y_i$.

The L2 loss decreases slowly compared to L1 loss when $a_i^L$ is close to $y_i$

The computational graph of L2 loss is given by,

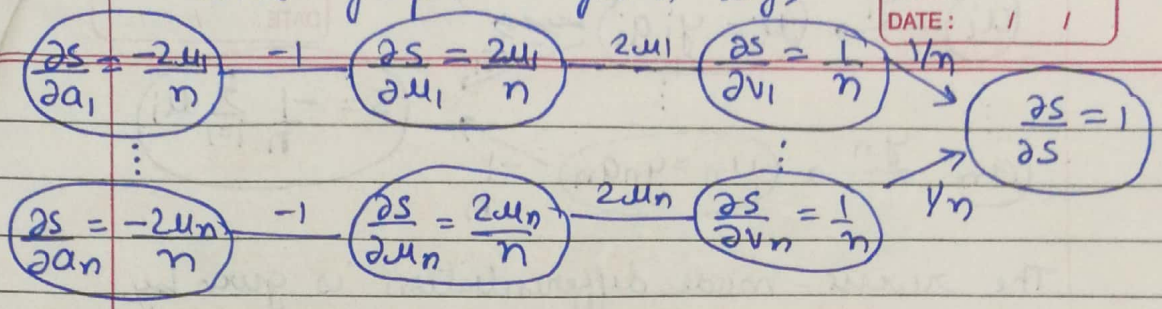



$$\frac{\partial u_i}{\partial a_i} = -1 \; ; \; \frac{\partial v_i}{\partial u_i} = 2u_i \; ; \; \frac{\partial S}{\partial v_i} = \frac{1}{n}$$

Here, $u_i = y_i - a_i^L$ ; $v_i = u_i^2$ ; $S = \frac{1}{n} \sum_{i=1}^{n} v_i$

are intermediate variables

The reverse-mode differentiation on the computational graph is given by,




We get, $\frac{\partial s}{\partial a_i} = -\frac{2u_i}{n} = -\frac{2}{n}(y_i - a_i^L) = \frac{2(a_i^L - y_i)}{n}$

This is same as normal differentiation.

The functional decomposition of L2 loss is given by,

$$C(a_1^L, \ldots, a_n^L) = S(v_1(u_1(a_1^L)), \ldots, v_n(u_n(a_n^L)))$$

6) Cosine similarity

The cosine similarity between two vectors $y$ and $\hat{y}$ is given by

$$\text{similarity} = \frac{y \cdot \hat{y}}{\|y\| \, \|\hat{y}\|}$$

Assume the final activations $a^L$ have been normalized. Also, $y_{target} = y$ is normalized. This is done to get clean results.

The loss function can be defined as,

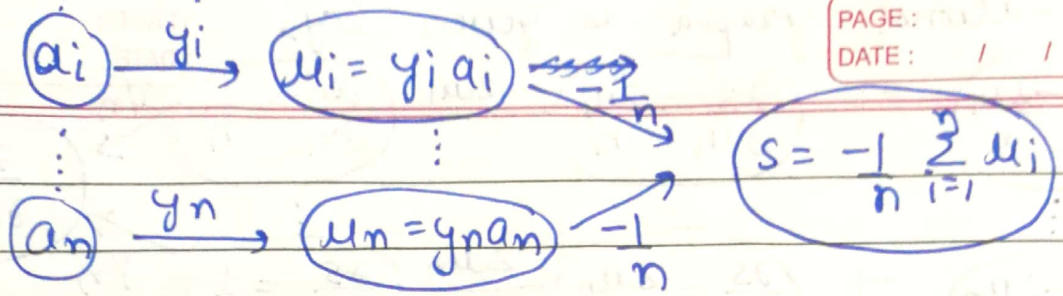
$$C = -\frac{1}{n} \sum_{i=1}^{n} y_i \cdot a_i^L$$

To maximize similarity & thus minimize the loss, a minus sign is added.

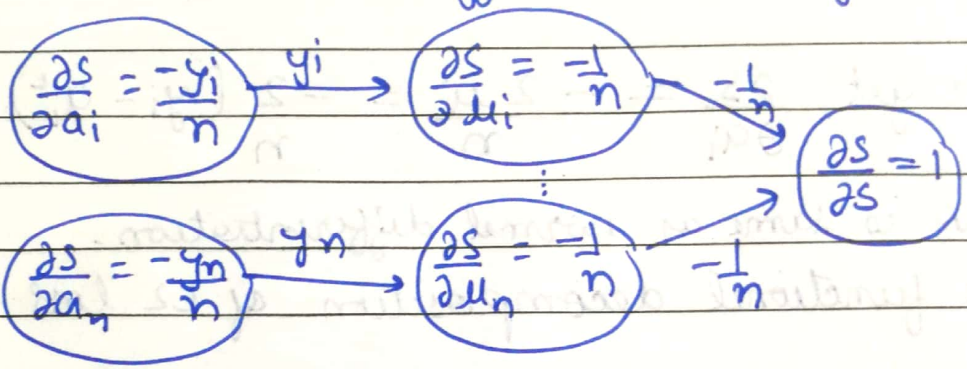
$$\boxed{\frac{\partial C}{\partial a_i^L} = -\frac{y_i}{n}}$$

Let $u_i = y_i a_i^L$ ; $S = -\frac{1}{n} \sum_{i=1}^{n} u_i$ be intermediate variables.

The computational graph is given by,




The reverse-mode differentiation is given by,




We get, $\frac{\partial s}{\partial a_i} = \boxed{\frac{\partial c}{\partial a_i} = -\frac{y_i}{n}}$ ,i.e., same as differentiation

The functional decomposition is given by,

$C(a_1^L, \ldots, a_n^L) = S(u_1(a_1^L), \ldots, u_n(a_n^L))$

---

Total 6 sheet bundle
Anubhav Agarwal
BE16B002