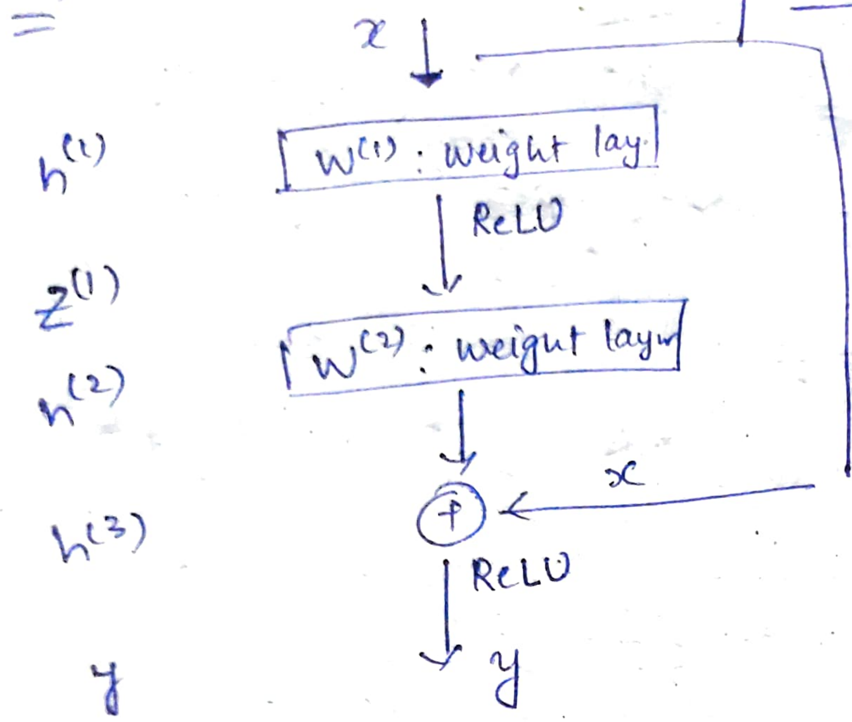
Assignment-03

K. Surya Prakash
EE18BTECH11026

Q1)




Forward pass:

o/p 1st blk: $h^{(1)} = x * w^{(1)}$

$z^{(1)} = \text{ReLU}(h^{(1)})$

o/p 2nd blk: $h^{(2)} = z^{(1)} * w^{(2)}$

$h^{(3)} = h^{(2)} + x$

$y = \text{ReLU}(h^{(3)})$

Note: $A \odot B \Rightarrow$ Element wise product which result in same dim

$A * B \Rightarrow$ Conv. to give same dim output

Assumtions:

1) $x$, $h^{(1)}$, $h^{(2)}$, $h^{(3)}$ all have the same dimensions

$\Rightarrow$ $\therefore$ The convolutions are done st, the sufficient padding is to be done.

$\Rightarrow$ Assuming $\frac{\partial L}{\partial y}$ is already known

# Backpropagation:

Need to find $\frac{\partial L}{\partial W^{(1)}}$; $\frac{\partial L}{\partial W^{(2)}}$; $\frac{\partial L}{\partial X}$

1) $$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial y} (\cdot) \left(h^{(3)} > 0\right) (\cdot) \underbrace{\frac{\partial h^{(3)}}{\partial h^{(2)}}}_{1} \cdot \frac{\partial h^{(2)}}{\partial W^{(2)}}$$

$$\frac{\partial L}{\partial W^{(2)}} = \left[\frac{\partial L}{\partial y} (\cdot) \left(h^{(3)} > 0\right)\right] 1 * \left(Z^{(1)}\right)$$

$$\boxed{\frac{\partial L}{\partial W^{(2)}} = \left(\frac{\partial L}{\partial y} (\cdot) \left(h^{(3)} > 0\right)\right) * Z^{(1)}}$$

$(\cdot)$ element wise product

$$\frac{\partial L}{\partial W^{(1)}} = \underbrace{\frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h^{(2)}} \cdot \frac{\partial h^{(2)}}{\partial z^{(1)}} \cdot (h^{(1)} > 0)}_{} \cdot \frac{\partial h^{(1)}}{\partial W^{(1)}}$$

$$\frac{\partial L}{\partial h^{(1)}} * X \longrightarrow (1)$$

$$\frac{\partial L}{\partial W^{(1)}} = \left[\left(\frac{\partial L}{\partial y} (\cdot) (h^{(3)} > 0)\right) * \left(W^{(2)}_{\text{flipped}}\right)\right] (\cdot) (h^{(1)} > 0) \Big] * X$$

$$\frac{\partial L}{\partial W^{(1)}} = \left\{\left[\left(\frac{\partial L}{\partial y} (\cdot) (h^{3} > 0)\right) * W^{(2)}_{\text{flip}}\right] (\cdot) (h^{(1)} > 0)\right\} * X$$

$$\boxed{\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial h^{(1)}} * X} \Bigg\} \text{Short form}$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} (\cdot) \frac{\partial y}{\partial h^{(3)}} (\cdot) \frac{\partial h^{(3)}}{\partial x}$$

$$= \frac{\partial L}{\partial y} (\cdot) ( h^{(3)} > 0 ) (\cdot) \frac{\partial h^{(3)}}{\partial x} \longrightarrow (ii)$$

* Computing $\frac{\partial h^{(3)}}{\partial x} = \frac{\partial h^{(2)}}{\partial x} + 1$ | 1 represent all one's matrix

$$\Rightarrow \frac{\partial h^{(2)}}{\partial x} = \left( * \, w^{(2)}_{flip} \right) (\cdot) \left( h^{(1)} > 0 \right) \left( * \, w^{(1)} \right)$$

$$\Rightarrow \frac{\partial h^{(2)}}{\partial x} = \left[ \left( * \, w^{(2)}_{flip} \right) (\cdot) ( h^{1} > 0 ) \right] * w^{(1)}$$

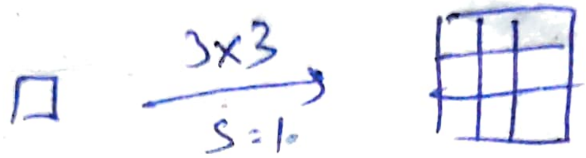
∴ Putting all together :

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial h^3} \left( 1 + \frac{\partial h^{(2)}}{\partial x} \right)$$

$$= \frac{\partial L}{\partial h^3} + \left[ \left( \frac{\partial L}{\partial h^3} * w^{(2)}_{flip} \right) (\cdot) ( h^{1} > 0 ) \right] * w^{(1)}_{flip}$$

Q2) 4 consecutive $3\times3$ conv layers

$\rightarrow$ stride: 1




For every layer each dimension of the support increases by '2'

$$\Rightarrow \quad 1\times1 \xrightarrow[C1]{} 3\times3 \xrightarrow[C2]{} 5\times5 \xrightarrow[C3]{} 7\times7 \xrightarrow[C4]{} 9\times9$$

$\therefore$ A single pixel in the image contributes to a $9\times9$ patch in the 4th non-image layer

$\Rightarrow$ Total effected pixels $= 81$

Q3)

→ If no. of hidden units → ↑

⇒ Model can learn more complex ~~rela~~

relations b/w O/P and I/P

⇒ But with more complexity

model has: less bias (Soln is closer to optimal solution)

: high variance

As model now learns complex relations, noise in the dataset is also captured, this results in fluctuations in the result

Model will be more sensitive

More hidden units } less bias
high variance

Q4) $y_k(x,w) = \sum w_{kj}^{(2)} \sigma\Big(\overbrace{\sum_i w_{ji}^{(1)} x_i + w_{jo}^{(1)}}^{z}\Big) + w_{ko}^{(2)}$

~~Let~~ we will be using tanh activation instead of sigmoid. & we desire to get same output : $y_K(x,w)$

* Definitions:

$\underbrace{w_{kj}^{(2)} ; w_{ji}^{(1)} ; w_{ko}^{(2)} ; w_{jo}^{(2)}}$

weights ~~of~~ with sigmoid model

$\underbrace{t_{kj}^{(2)} ; t_{ji}^{(1)} ; t_{ko}^{(2)} ; t_{jo}^{(2)}}$

weights with tanh model.

* Relation b/w tanh & sigmoid

* $\tanh(z) = \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}} = \dfrac{2}{1+e^{-2z}} - 1$

$= 2\sigma(2z) - 1$

$\sigma(z) = 1/1+e^{-z}$ //

Let $z_j = \sum_i w_{ji}^{(1)} x_i + w_{j0}^{(1)}$ } with Sigmoid weights

$a_j = \sum_i t_{ji}^{(1)} x_i + t_{j0}^{(1)}$ } with tanh weights.

* For Sigmoid model:

$$y_k(x,w) = \sum w_{kj}^{(2)} \sigma(z_j) + w_{k0}^{(2)} \longrightarrow (1)$$

$$y_k(x,t) = \sum t_{kj}^{(2)} \tanh(a_j) + t_{j0}^{(2)}$$

$$= \sum \left(2t_{kj}^{(2)} \cdot \sigma(2a_j) - 1\right) + t_{j0}^{(2)}$$

$$= \sum_j 2t_{kj}^{(2)} \sigma(2a_j) + t_{j0}^{(2)} - \sum 2t_{kj}^{(2)} \longrightarrow (2)$$

Comparing (1) & (2)

$$2a_j^{i} = z_j^{i} \longrightarrow (i)$$

$$\Rightarrow \sum 2t_{ji}^{(1)} x_i^{i} + 2t_{jo}^{(1)} = \sum w_{ji}^{(1)} x_i^{i} + w_{jo}^{(1)}$$

By comparing:

$$\left.\begin{array}{l} \boxed{t_{ji}^{(1)} = \frac{w_{ji}^{(1)}}{2}} \\ \boxed{t_{jo}^{(0)} = \frac{w_{jo}^{(0)}}{2}} \\ \text{Also. } \boxed{t_{kj}^{(2)} = \frac{w_{kj}^{(2)}}{2}} \\ \boxed{t_{ko}^{(2)} - \sum 2t_{kj}^{(2)} = w_{ko}^{(2)}} \end{array}\right\} \text{Linear transformations}$$

∴ By making the above transformations to the existing weights we can obtain the same output even by using tanh activation. Hence she is correct//

Q5) Quadratic error:

$$E(w) \approx E(w^*) + \frac{1}{2}(w-w^*)^T H (w-w^*)$$

H: Hesian evaluated @ $w^*$

* we know that, if $w^*$ is the optimal sol'n

H is a PSD

⇒ Eigen values are:

$$Hu_i = \lambda_i u_i \quad (\lambda_i \geq 0 \because H: \text{PSD})$$

where,

$u_i$: Eigen vectors of H, s.t

$u_i$ are orthonormal

$$\Rightarrow \boxed{u_i^T u_j = \delta_{ij}} \rightarrow (1)$$

* Let us express $w-w^*$ as linear combination of eigen vectors

$$\boxed{w-w^* = \sum_i \alpha_i u_i} \rightarrow (2)$$

$$\therefore \frac{1}{2}(w-w^*)^T H (w-w^*) = \left(\frac{1}{2}\right)\left(\sum \alpha_i u_i\right)^T H \left(\sum_j \alpha_j u_j\right)$$

$$= \frac{1}{2}\left(\sum_i \alpha_i u_i\right)^T \cdot \left(\sum \alpha_j H u_j\right)$$

$$= \frac{1}{2}\left(\sum_i \alpha_i u_i\right)^T \left(\sum_j \alpha_j \lambda_j u_j\right)$$

$$= \frac{1}{2} \sum_i \sum_j (\lambda_j)\left(\alpha_i \alpha_j u_i^T u_j\right)$$

* we know that from (1)

$$u_i^T u_j = \delta_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

$$= \frac{1}{2} \sum_i \lambda_i \alpha_i^2$$

$$\therefore \underbrace{E(w) - E(w^*)} \simeq \frac{1}{2} \sum_i \lambda_i \alpha_i^2$$

$\Rightarrow$ Since 'H' is a PSD $\Rightarrow$ $E(w) < Et$.

$$v^T H v \geq 0$$

$\rightarrow$ $E(w) - E(w^*) \geq 0$

$$\Rightarrow \boxed{\frac{1}{2} \sum_i \lambda_i \alpha_i^2 \geq 0}$$

$\Rightarrow$ This can be written in the form:

$$\sum \lambda_i d_i^2 = c^2 ; \quad c^2 = 2(E(w) - E(w^*))$$

* For a constant error : $E(w)$ = constant

$\Rightarrow$ $c^2$ = constant

$\because$ $w^*$ : fixed vector

$\therefore$ $\boxed{\sum_i \lambda_i d_i^2 = c^2}$ : (positive constant)

↳ This represents elliptical contours

where axes are $u_i$ (eigen vectors of H)

$$\Rightarrow \quad w = w^* + \sum d_i u_i$$

↳ $w - w^* = \sum d_i u_i$

~~A trans~~

Translation of axes from origin as center to $w^*$ as center.

* The length of axis : $\alpha_j^i$ is computed by setting $\left.\alpha^i\right|_{i \neq j} = 0$.

$$\Rightarrow \boxed{\alpha_j^i = \sqrt{\frac{c^2}{\lambda_j^i}}};$$

we know that $\lambda_j^i$ : eigen value of $H \geq 0$.

$$\therefore \left( \alpha_j \propto \frac{1}{\sqrt{\lambda_j^i}} \right) //.$$

* length of axis is inversely proportional to square root of corresponding eigen value $(\lambda_j^i)$

Q6) We can use 'transfer learning'

Available dataset :

20 images of 200 classes

⇒ Comment: Small dataset

Available deployed model.

→ Trained on large dataset.

~~Dataset~~
Trained data is similar to available data.

↳ We can borrow the deployed model specifi-cations (architecture, pre-trained weights) and retrain by making some tweaks to the model.

## General understanding :

1) Freeze the 'Source task generic layers' & 'Source task special layers'.

⇒ These layers are responsible for feature extraction. Since both data are alike the features are likely to be same

2) Discard the existing classification layer (which classifies 1000 classes).

3) Attatch a new classification layer (which classifies 200 classes), with random initialise weights.

4) Train the network, with the available data, make sure weight updation should be done in the newly attatched 'classification layer'

* The final layer should have 200 ~~classes~~ neurons to classify 200 classes.