Deep learning

Name: Siddharth Pansuria

CWID: 20005837

Mid term

1)

a) False

b) True

c) True

d) True

e) False

2)

a) A

b) B

c) B, C, D

d) E

e) A

f) A, B, D

3)

Linear function cannot be used because it would simply map input to output like linear regression and no feature maps would be generated. It would not be accurate as we need it to be

CUID : 20005837

3)

a) $J = \frac{1}{2}(x_1^2 x_2 - y)^2$

$\frac{\partial J}{\partial x_1} = \frac{1}{2} \cdot 2(x_1^2 x_2 - y) \cdot 2x_1 x_2$

$\therefore \frac{\partial J}{\partial x_1} = 2x_1 x_2 (x_1^2 x_2 - y)$

$\boxed{\text{Ans} = 2x_1 x_2 (x_1^2 x_2 - y)}$

b) $J = \frac{1}{2}(x_1^2 x_2 - y)^2$

$\frac{\partial^2 J}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_1}\left(\frac{\partial J}{\partial x_2}\right)$

$\therefore \frac{\partial J}{\partial x_2} = \frac{1}{2} \cdot 2(x_1^2 x_2 - y) \cdot x_1^2$

$= x_1^2 (x_1^2 x_2 - y)$

$= x_1^4 x_2 - y x_1^2$

$\therefore \frac{\partial}{\partial x_1}\left(\frac{\partial J}{\partial x_2}\right) = 4x_1^3 x_2 - 2x_1 y$

$\therefore \boxed{\text{Ans} = (4x_1^2 x_2 - 2y) \cdot x_1}$

CUID : 20005837

c) $J = \frac{1}{2}(x_1^2 x_2 - y)^2$

Now let x be a vector,

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\therefore \frac{\partial J}{\partial x} = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \end{bmatrix}$$

$$\frac{\partial J}{\partial x_1} = \frac{1}{2} \cdot 2(x_1^2 x_2 - y) \cdot 2x_1 x_2$$

$$= 2x_1 x_2 (x_1^2 x_2 - y)$$

$$\frac{\partial J}{\partial x_2} = \frac{1}{2} \cdot 2(x_1^2 x_2 - y) \cdot x_1^2$$

$$= x_1^2 (x_1^2 x_2 - y)$$

$$\therefore \frac{\partial J}{\partial x} = \begin{bmatrix} 2x_1 x_2 (x_1^2 x_2 - y) \\ x_1^2 (x_1^2 x_2 - y) \end{bmatrix}$$

$$\boxed{\text{Ans} = \begin{bmatrix} 2x_1 x_2 (x_1^2 x_2 - y) \\ x_1^2 (x_1^2 x_2 - y) \end{bmatrix}}$$

CWID : 20005837

d) $J = \frac{1}{2}(x_1^2 x_2 - y)^2$

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $\quad x^T = [x_1 \quad x_2]$ $\quad (p \times q)$
(1×2)

Let z = reshape (x^T) to $pq \times 1$

$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$$\frac{\partial J}{\partial z} = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 x_2 (x_1^2 x_2 - y) \\ x_1^2 (x_1^2 x_2 - y) \end{bmatrix}$$

reshape $\frac{\partial J}{\partial z}$ back to $p \times q$ to get back $\frac{\partial J}{\partial x^T}$.

$$\therefore \frac{\partial J}{\partial x^T} = [2x_1 x_2 (x_1^2 x_2 - y) \quad x_1^2 (x_1^2 x_2 - y)]$$

let us again reshape $\frac{\partial J}{\partial x^T}$ to $pq \times 1$

$\therefore z'$ = reshape $\frac{\partial J}{\partial x^T}$ to $pq \times 1$

$$= \begin{bmatrix} 2x_1 x_2 (x_1^2 x_2 - y) \\ x_1^2 (x_1^2 x_2 - y) \end{bmatrix}$$

CWID : 20005897

suppose $z' = \begin{bmatrix} z_1' \\ z_2' \end{bmatrix}$

$$\therefore \frac{\partial z'}{\partial x} = \begin{bmatrix} \frac{\partial z_1'}{\partial x_1} & \frac{\partial z_1'}{\partial x_2} \\ \frac{\partial z_2'}{\partial x_1} & \frac{\partial z_2'}{\partial x_2} \end{bmatrix}$$

$$\therefore \frac{\partial z'}{\partial x} = \begin{bmatrix} 6x_1^2 x_2^2 - 2x_2 y & 4x_1^3 x_2 - 2x_1 y \\ 4x_1^3 x_2 - 2x_1 y & x_1^4 \end{bmatrix}$$

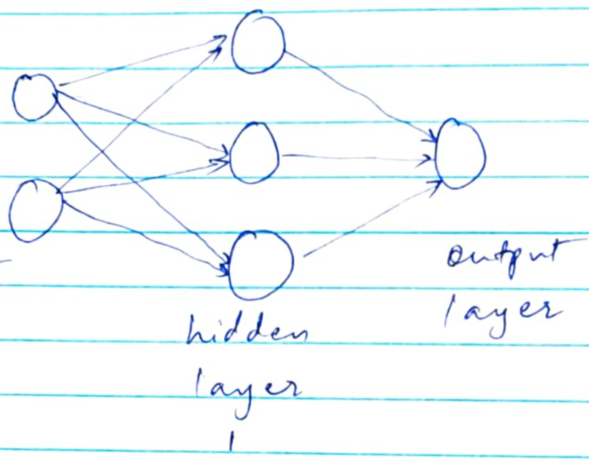
Now reshaping it to original shape

$$\therefore \frac{\partial}{\partial x}\left(\frac{\partial J}{\partial x^T}\right) = \begin{bmatrix} 6x_1^2 x_2^2 - 2x_2 y & 4x_1^3 x_2 - 2x_1 y \\ 4x_1^3 x_2 - 2x_1 y & x_1^4 \end{bmatrix}$$

$$\boxed{\text{Ans} = \begin{bmatrix} 6x_1^2 x_2^2 - 2x_2 y & 4x_1^3 x_2 - 2x_1 y \\ 4x_1^3 x_2 - 2x_1 y & x_1^4 \end{bmatrix}}$$

CWID : 20005837

4)



a) from input layer to hidden layer 1 : $g = Vx$

applying activation function to vector : $h = \max\{0, g\}$

from hidden layer to output layer : $\hat{y} = Wh$

b) $L = \frac{1}{2}(y - \hat{y})^2$

$$\frac{\partial L}{\partial W} = \frac{\partial}{\partial W}\left(\frac{1}{2}(y-\hat{y})^2\right) = \frac{\partial}{\partial W}\left(\frac{1}{2}(y - Wh)^2\right)$$

$$= \frac{1}{2} \cdot 2(y - Wh)\cdot(-h) = h\cdot(Wh - y)$$

$$\boxed{\text{Ans} = h\cdot(Wh - y)}$$

CWID: 20005837

c) $L = \frac{1}{2}(y-\hat{y})^2$

$$\frac{\partial L}{\partial v} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h} \cdot \overbrace{\frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial v}}^{\frac{\partial h}{\partial v}}$$ (using chain rule)

$$= (\hat{y}-y) \cdot \frac{\partial (wh)}{\partial h} \cdot \frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial v}$$

$$= (\hat{y}-y) \cdot w \cdot \frac{\partial h}{\partial g} \frac{\partial g}{\partial v}$$

$$= (\hat{y}-y) \cdot w \cdot \begin{pmatrix} v & , \text{if } g > 0 \\ 0 & , \text{else} \end{pmatrix}$$

$$= \begin{cases} (\hat{y}-y) \cdot w \cdot v & \text{if } g > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\therefore \boxed{\frac{\partial L}{\partial v} = \begin{cases} (\hat{y}-y) \cdot w \cdot v & \text{if } g > 0 \\ 0 & \text{otherwise} \end{cases}}$$

d) For stochastic gradient descent, just select a random j from the n training samples. For v, compute stochastic gradient $sgd_j = \frac{\partial L_j}{\partial v}$ at $v = v_{old}$, then update, $v_{new} = v_{old} - \eta \cdot sgd_j$. Do the same for w in each iteration. So for each iteration,

$sgd_{j,v} = \left.\frac{\partial L_j}{\partial v}\right|_{v=v_{old}}$, $sgd_{j,w} = \left.\frac{\partial L_j}{\partial w}\right|_{w=w_{old}}$, $v_{new} = v_{old} - \eta\, sgd_{j,v}$

$w_{new} = w_{old} - \eta\, sgd_{j,w}$

CWID : 20005837

5)

Layers	Output shape	No. of parameters
Input	[32, 32, 3]	0
CONV(10, 5)	[28, 28, 10]	$((5\times5\times3)+1)\times10$
POOL(2)	[14, 14, 10]	0
CONV(5, 3)	[12, 12, 5]	$((3\times3\times10)+1)\times5$
POOL(2)	[6, 6, 5]	0
FC(10)	[10, 1]	$10\times(6\times6\times5+1)$

GWID : 20005837

6)

a) Main difference :

→ For full batch descent, we can say that the batch-size always remains the same ie n. For mini-batch descent, the batch size will always remain non zero and greater than 1 but less than the full sample size. For stochastic batch size is always 1 since we only take one random sample

→ The main advantage of using mini-batch GD instead of full batch GD is that the cost of an iteration is significantly smaller for mini-batch and it also converges quicker as compared to full batch GD.

b) Each layer that is implemented in a deep neural network would simply map the input linearly to the output, if non-linear activation functions aren't used. So the network wouldn't be expressive and it wouldn't actually produce good or proper results. Thus it is necessary to implement non-linear activation function in deep neural network otherwise it would just be similar to linear regression.

CWID : 20005837

c)

(i) Data Augmentation :

→ This method simply generates more training samples from existing training data. eg. using flip, rotation, etc.
→ It can only be used for training samples and not for validation and test samples.
→ It is a method to generate a slightly newer data without any extra cost.

(ii) Pre-train :

→ We can pre-train our deep net on a large dataset.
→ We can remove top layers and we can also build new top layers.
→ We can also fine-tune the top layers or we can just keep the base layer same and train the top layers.

d) To tune, we can use k-fold cross validation:
→ Propose a grid of hyper parameters.
→ Randomly partition training samples to k parts (k-1 : training 1: ~~test~~ validation)
→ Compute average test errors of the k repeats. (average is validation error)
→ Choose the hyper-parameter leading to the smallest validation error.