Q1) $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots \beta_d x_{id}$

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} \cdots & x_{nd} \end{bmatrix}$$

- $\hat{y} = X\beta$
- $J(\beta) = \frac{1}{2} \| y - X\beta \|^2$

$$J(\beta) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

* Normal eqn's obtained by solving

$$\frac{\partial J(\beta)}{\partial \beta} = 0$$

(a) Regression's function?

Regression learns a linear relationship b/w input variables x and output y by estimating parameters β that best predicts y.

Why Squared error is minimized?

- It effects large errors heavily
- Differentable and convex
- Leads to a closed form soln. → sq. error represents diff. b/w predicted and actual values of y. minimising $\uparrow$ accuracy

[* close to reality as possible ↓]

(b) Derivation Of the Normal Eqⁿ :-

$$J(\beta) = \frac{1}{2}(y - x\beta)^T (y - x\beta)$$

$$= \frac{1}{2}\left(y^T y - 2y^T x\beta + \beta^T x^T x\beta\right)$$

Taking gradient wrt β :-

$$\nabla_\beta J = -x^T y + x^T x\beta$$

Setting gradient to '0' :-

$$x^T x\beta = x^T y$$

Gives Normal Eqⁿ :-

$$\beta = (x^T x)^{-1} x^T y$$

(c) Why direct $x^T x$ inversion is problematic?

$\beta = (x^T x)^{-1} x^T y$ → issues :-

① Computational cost
- forming :- $x^T x : O(nd^2)$
- Inverting :- $x^T x : O(d^3)$

② Memory issues :-
- Requires loading entire dataset into memory.

③ Numerical Instability.
$x^T x$ can be Singular

∴ Iterative mtds like gradient descent are preferred for large datasets

Q2) Back Propagation & Gradient :-

(a) Computes efficiently how much each parameter (wt's & bias) contributes to final predict^{n} error & updates them to reduce error.

Works by :-

- Propagating inputs fwd thru network
- Computing loss @ output.
- Propagating gradients backward using chain rule.

(allows gradients to be computed layer-by layer)

↓

Loss > output > Hidden layers > Parameters

(b) Gradient computat^{n} for 2 layer Network :-

$$z_1 = w_1 x + b_1 \; ; \quad a_1 = \sigma(z_1)$$
$$z_2 = w_2 a_1 + b_2 \; ; \quad a_2 = \sigma(z_2) = \hat{y}$$

Sigmoid activat^{n} :-

$$\sigma(z) = \frac{1}{1+e^{-z}} \; ; \quad \sigma'(z) = \sigma(z)\left(1-\sigma(z)\right)$$

Binary cross entropy loss :-

$$L = -\left[y \log(a_2) + (1-y)\log(1-a_2)\right]$$

Gradient wrt w_2 :-

$$\frac{\partial L}{\partial w_2} = (a_2 - y)\, a_1$$

wrt b_2 :-

$$\frac{\partial L}{\partial b_2} = (a_2 - y)$$

wrt w_1 :-

$$\frac{\partial L}{\partial w_1} = (a_2 - y) w_2 \, \sigma'(z_1) \, x$$

w.r.t b_1 :

$$\frac{\partial L}{\partial b_1} = (a_2 - y)\sigma(z_1)$$

(c) Gradient Descent Update Rule :-

For any parameter θ :-

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{\partial L}{\partial \theta}$$

Applied Parameters :-

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

$$b_1 \leftarrow b_1 - \eta \frac{\partial L}{\partial b_1}$$

$$w_2 \leftarrow w_2 - \eta \frac{\partial L}{\partial w_2}$$

$$b_2 \leftarrow b_2 - \eta \frac{\partial L}{\partial b_2}$$

Role of learning rate η :-

- Too large → updates overshoot → divergence
- Too small → very slow convergence.
- Proper choice → Stable & efficient training.

(4) LSTMs in Natural Language Processing

(a) Example of long-range dependency in natural lang:-

"The book I recommended Dhrishti was fascinating."

To understand "was fascinating" the model must remember "the book" which appears earlier.

- A std. RNN would struggle due to vanishing gradients
- An LSTM can retain the info using memory cell.

(b) • The LSTM Memory cells store imp. info across time steps,

- Forget gate decides what info to erase.
- Input gate controls new info to add.
- Output gate controls what part of memory influences the output.

Ex:- Machine Translation scenario for forget gate ≈ 0

Consider Translating:-

"I went to the bank. Later that day, I deposited the money".

- When the word "bank" appears again in a financial context, forget gate suppresses irrelevant meanings.

This allows the LSTM to discard outdated context & focus on the correct interpretation.

Q3) ANN vs RNN vs LSTM

(a) ANNs:-

- Process inputs independently
- Each input sample is treated as a fixed size vector.
- No memory of prev. inputs
- Suitable for problems where data pts are unrelted.

RNNs:-

- Process sequential data one step @ a time
- Maintain a hidden state that carries info from prev. time steps.
- Output @ time t depends on both current input & past inputs

(b) Simple RNNs struggle w/ long term Dependencies :-

- RNNs are trained using BPTT.
  ↓
  Gradients are repeatedly multiplied by wt. across time steps.
  ↓
  leads to :- ① Vanishing Gradients (become very small)
  ② Exploding Gradients (Grow uncontrollably)

(c) Role of Gates in LSTMs :-

LSTMs introduce gating mechanisms to control info flow:

- Forget Gate : decides what info to discard.
- Input Gate : decides what new info to store.
- Output gate : decides what info to expose to next layer

(d) How LSTMs address the vanishing gradient problem?

- LSTMs use a cell st. that allows info to flow w/ minimal modification".
  ↓
  Additive updates preserve gradients
  ↓
  Gates regulate updates, preventing unnecessary shrinking
  ↓
  Enables long range dependencies

(e) Extask :-

- ANN :- House price predict" using tabular features
- RNN :- Part of speech tagging in short sentences
- LSTM :- Speech recognit" or language translat" & long sentences.