ASSIGNMENT - 1

Q1) (a) Linear regression is a statistical method used to model the relationship between a dependent variable (target) and one or more independent variables (features). Its goal is to find the optimal set of parameters (weights) that form a linear equation best predicting the output for a given input.

Minimizing the squared error as a cost fn. is a good choice due to the following reasons:

1) Penalizing large errors - (by squaring the diff)
2) Mathematical Convenience - is convex and cont. diff. This ensures that the global min. can be found efficiently using calculus.

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

$\|z\|^2 = z^T z$

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

$[(A-B)^T = A^T - B^T]$

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

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

↓ scalar

$$y^T X\beta = (y^T X\beta)^T = \beta^T X^T y$$

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

$$\frac{\partial J(\beta)}{\partial \beta} = \frac{1}{2}(0 - 2X^T y + 2X^T X\beta)$$

$$= -X^T y + X^T X\beta$$

$$\frac{\partial J(\beta)}{\partial \beta} = 0 \Rightarrow \boxed{X^T X\beta = X^T y} \text{ normal eqn}$$

(c) Problematic due to the following reasons:

- Computational Complexity:
  Matrix inversion scales at roughly $O(d^3)$. As the number of features (d) increases, the calculation time grows cubically (very slow)
- Memory constraints: Storing the $X^T X$ matrix requires $O(d^2)$ memory, which can exceed RAM for large data.
- Singularity: If features are highly correlated (multicollinearity), the matrix becomes singular, making the direct calculation impossible.

iterative methods like gradient descent are preferred because they avoid matrix inversion entirely. They are much more efficient computationally (often linear complexity per step), require less memory and can find approximations even when the matrix is singular.

9. (a) Backpropagation efficiently calculates the gradient of the loss function L with respect to every weight and bias in the network. It works by calculating the error at the output and propagating it ~~backprop~~ backwards to the input layer, telling the network exactly how to adjust parameters to minimize error

Efficiency of the chain rule -

Since neural networks are nested fns, the chain rule is used to find their derivatives. Backpropagation applies the Chain rule starting from the last layer and working backward. This allows the algorithm to reuse gradients computed for later layers to solve for earlier ones, avoiding redundant calculations and making training computationally efficient.

(b) Gradients for output layer (w_2, b_2)

$$\frac{\partial L}{\partial z_2} = a_2 - y$$

$$\delta_2 = a_2 - y$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z_2} \times \frac{\partial z_2}{\partial w_2}$$

$$z_2 = w_2 a_1 + b_2$$

$$\frac{\partial z_2}{\partial w_2} = a_1$$

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

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2}$$

$$\frac{\partial z_2}{\partial b_2} = 1$$

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

Gradients for the hidden layer (w_1, b_1)

$$\frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1}$$

$\downarrow$ $a_2 - y$; $\downarrow$ w_2 ; $\downarrow$ $\sigma'(z_1) = a_1(1 - a_1)$ [$a_1 = \sigma(z_1)$ ↓ sigmoid]

$$\frac{\partial L}{\partial z_1} = (a_2 - y) w_2 \cdot a_1 (1 - a_1)$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}$$

$$= [(a_2 - y) w_2 \cdot a_1 (1 - a_1)] \cdot x$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1}$$

$$\frac{\partial L}{\partial b_1} = (a_2 - y) \cdot w_2 \cdot a_1 (1 - a_1)$$

(c) In gradient descent, we update the parameters by moving them in the opposite direction of the gradient to minimize the loss.

Let $\eta \longrightarrow$ learning rate

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

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

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

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

The learning rate (η) is a hyperparameter that controls the size of the step we take during the update.

- If η is too small - The network will run slowly (many iterations to converge)
- If η is too large - Might overshoot the min, causing the loss to diverge or oscillate without converging.

5. (a) Input processing ANNs process inputs simultaneously as independent, fixed size vectors, meaning they have no sense of order or time. RNNs process inputs sequentially maintaining a hidden state memory that carries info from previous steps to the current one.

(b) Simple RNNs suffer from the vanishing gradient problem. During training, gradients are repeatedly multiplied by weights. Over long sequences, these gradients shrink to zero, causing the network to effectively stop learning from early inputs and forget long-term context.

(c) LSTMs use 3 gates (Forget, Input, Output) to control what info is kept or thrown away. By selectively updating a protected "Cell state", these gates allow the network to hold onto important info for long periods while discarding irrelevant noise.

(d) LSTMs allow gradients to flow backward through the cell state using addition rather than just multiplication. This creates a direct path (often called "const error carousel") for the error signal to travel back to

the beginning of the sequence without decaying, allowing the model to learn long-term dependencies.

(c) Example tasks

- ANN : House price prediction (tabular data)
- RNN : short-term weather forecasting (simple sequences)
- LSTM : Language translation (long, complex sequ)

4. (a) Ex.- Subject verb agreement across a long gap : "The flights (plural), which were delayed due to bad weather..., were cancelled".

The model must remember "flights" is plural to predict "were" instead of "was.".

Standard RNN would struggle due to vanishing gradient problem (might forget)

(b) • Intuition : The memory cell acts like a conveyor belt, carrying data unchanged. Gates add or remove info. Since gradients flow via addition (avoiding repeated multiplication), memories persist without fading.

• Forget gate scenario - The gate closes (output ≈ 0) at a sentence boundary. This erases the old context so the model starts fresh for the new sentence.