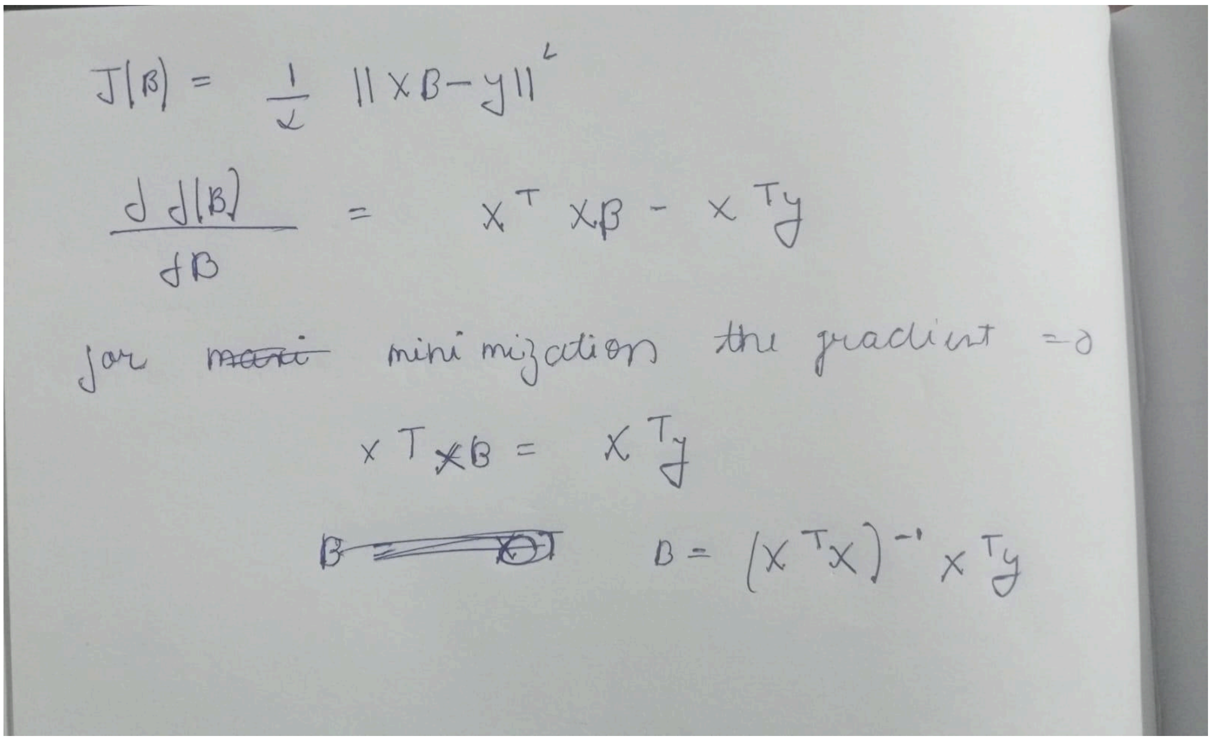
q1) (a) What regression does and why squared error is minimized

Regression is used to predict a continuous output by finding a relationship between input variables and the target variable. It finds the best-fitting line (or function) that minimizes the difference between predicted and actual values.

Squared error is minimized because it:

- Penalizes large errors more heavily
- Gives a single optimal solution
- Makes the problem mathematically simple and stable

(b) Derivation of the normal equation



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

$$\frac{d\, J(\beta)}{d\beta} = X^T X\beta - X^T y$$

for ~~maxi~~ minimization the gradient $= 0$

$$X^T \cancel{X}\beta = X^T y$$

~~$B =$~~ [illegible] $\quad B = (X^T X)^{-1} X^T y$

(c) Why direct inversion is problematic & why iterative methods are preferred

Direct inversion of XTXX^T XXTX is problematic because:

- It is computationally expensive for large datasets

- XTXX^T XXTX may be singular or ill-conditioned
- It requires high memory

Iterative methods like gradient descent are preferred because:

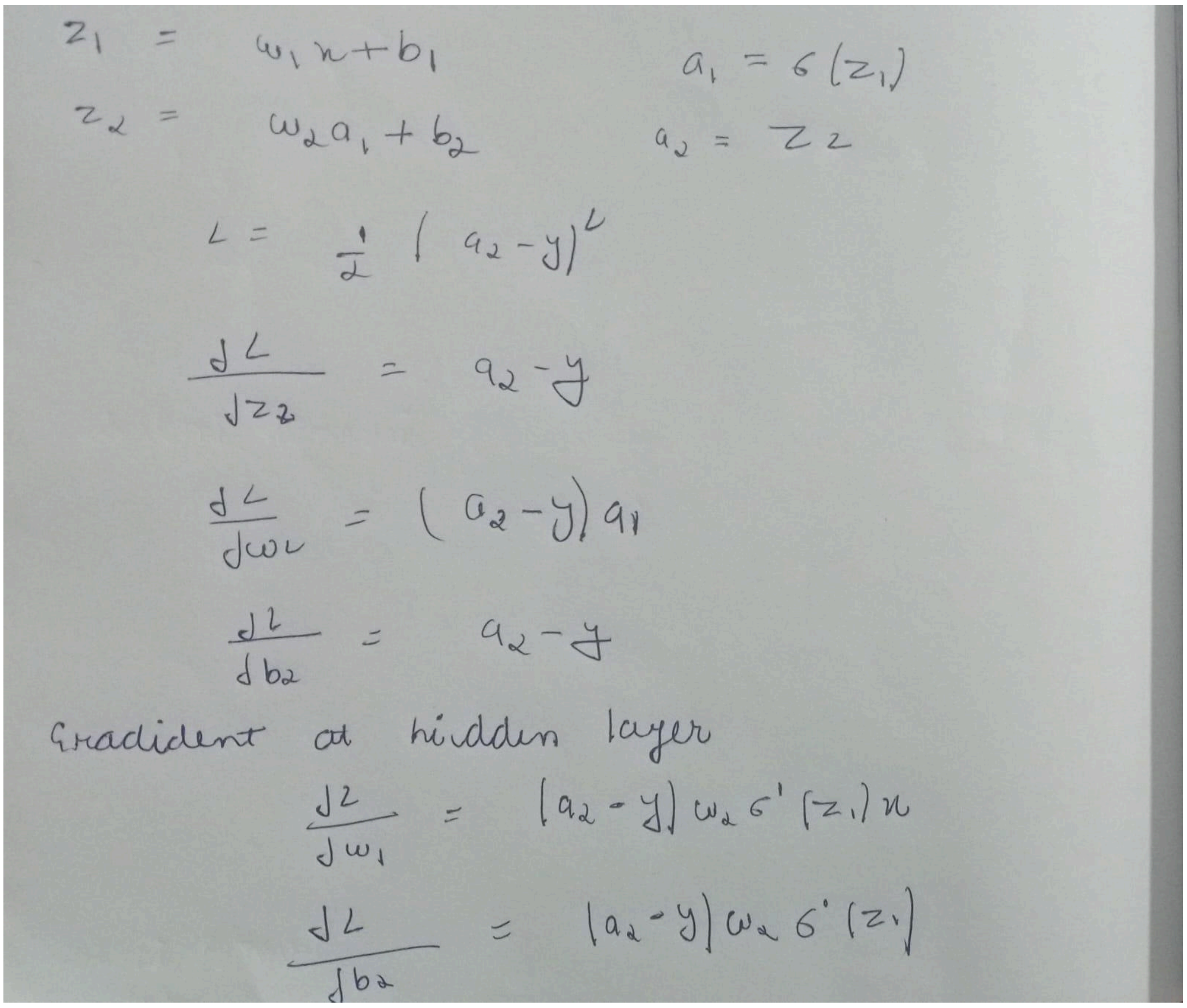
- They avoid matrix inversion
- Scale well for large and high-dimensional data
- Are computationally efficient

q2) (a) Backpropagation and chain rule

Backpropagation is a method used to calculate how much each weight and bias contributes to the final error in a neural network. It works by sending the error from the output layer back to the earlier layers.

The chain rule helps by breaking a complex calculation into smaller steps, so gradients can be computed efficiently layer by layer.

(b) Gradients



$$z_1 = w_1 x + b_1 \qquad a_1 = \sigma(z_1)$$

$$z_2 = w_2 a_1 + b_2 \qquad a_2 = z_2$$

$$L = \frac{1}{2}(a_2 - y)^2$$

$$\frac{dL}{dz_2} = a_2 - y$$

$$\frac{dL}{dw_2} = (a_2 - y)\, a_1$$

$$\frac{dL}{db_2} = a_2 - y$$

Gradient at hidden layer

$$\frac{dL}{dw_1} = (a_2 - y)\, w_2\, \sigma'(z_1)\, x$$

$$\frac{dL}{db_2} = (a_2 - y)\, w_2\, \sigma'(z_1)$$

(c) Gradient descent update and learning rate

Weights and biases are updated as:

new value=old value−η×gradient\text{new value} = \text{old value} - \eta \times \text{gradient}new value=old value−η×gradient

The learning rate decides how big the update step is.
A large learning rate may cause instability, while a small one makes learning slow.

Q3: ANN vs RNN vs LSTM

(a) ANN vs RNN input processing
ANN processes inputs independently and does not remember past inputs.
RNN processes data sequentially and keeps information from previous steps using a hidden state.

(b) Why simple RNNs struggle with long-term dependencies
In long sequences, gradients become very small during training.
Because of this, RNNs forget information from earlier time steps, known as the vanishing gradient problem.

(c) Role of gates in LSTMs
LSTMs use input, forget, and output gates.
These gates control what information to keep, update, or discard, helping the model remember useful information.

(d) How LSTMs address the vanishing gradient problem
LSTMs use a memory cell that allows information and gradients to flow over long sequences.
This helps preserve long-term dependencies.

(e) One example task for each model
ANN: House price prediction
RNN: Next word prediction
LSTM: Machine translation or sentiment analysis

Q4: LSTMs in Natural Language Processing

(a) Example of long-range dependency
Example sentence:
“The book that you gave me yesterday is very interesting.”

The verb “is” depends on the word “book,” which appears much earlier.
A standard RNN would struggle to capture this, while an LSTM can handle it.

(b) Memory cell, gates, and forget gate example
The memory cell stores important information for a long time.
Gates decide what information should be kept or forgotten.

In machine translation, when the sentence topic changes, the forget gate should remove old context so the new sentence is translated correctly.