~~a) Regression is essentially a way to draw a line through the data that best predicts the output y from the input features x.~~

a) Regression fits a line & hyperplane to predict outputs y from input features x. It tries to find the best parameters β that maps inputs to outputs. The quality of the fit is measured using the squared error, $\sum(y_i - \hat{y}_i)^2$. Squaring ensures both positive and negative errors are treated equally & penalizes large mistakes more. The factor ½ is for mathematical convenience when differentiating. Minimizing this error gives the parameters that make predictions as close as possible to actual values.

b) Cost function,

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

$$\|y - X\beta\|^2 = (y - X\beta)^T(y - X\beta)$$

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

$$\Rightarrow \nabla_\beta J(\beta) = -X^T(y - X\beta)$$

To minimize the cost,

$$-X^T(y - X\beta) = 0$$

$$X^Ty - X^TX\beta = 0 \Rightarrow X^TX\beta = X^Ty \Rightarrow \boxed{\beta = (X^TX)^{-1}X^Ty}$$

c)
Directly inverting X^TX can be slow & unstable for large & high-dimensional datasets. If some features are highly corelated, X^TX may be nearly singular, causing errors in β. It also requires a lot of memory to store & invert the matrix. Gradient descent avoids these problems by updating β step by step. Without needing inversion. It works well on large datasets & is more stable when the data is ill-conditioned. You can think of it as walking down the hill gradually instead of trying to teleport to the bottom.

(2)
a) Backpropagation is a method to calculate how the loss changes with each weight and bias in a neural network. It uses chain rule to break down derivatives into small, mangeable parts. This makes it efficient as we can reuse values from forward pass. Essentially, it propagate the error backward through the network so we know how to adjust each parameter to reduce the loss.

b) $z_1 = w_1 x + b_1, \quad a_1 = \sigma(z_1)$

Forward pass: $z_2 = w_2 a_1 + b_2, \quad a_2 = \sigma(z_2)$

Loss: $L = -(y \log a_2 + (1-y) \log(1-a_2))$

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

$$\Rightarrow \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z_2} \frac{\partial z_2}{\partial w_2} = (a_2 - y) \cdot a_1$$

$$\Rightarrow \frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2} \frac{\partial z_2}{\partial b_2} = (a_2 - y) \cdot 1 = a_2 - y$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}$$

$$\Rightarrow \frac{\partial L}{\partial w_1} = (a_2 - y) w_2 \, a_1(1-a_1) \cdot x$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1}$$

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

c) we update each parameter using the gradient:

$$w_1 - \eta \frac{\partial L}{\partial w_1} \to w_1 \qquad b_1 \leftarrow b_1 - \eta \frac{\partial L}{\partial b_1}$$

$$w_2 - \eta \frac{\partial L}{\partial w_2} \to w_2 \qquad b_2 \leftarrow b_2 - \eta \frac{\partial L}{\partial b_2}$$

* η = learning rate, controls the step size
- Too large: may overshoot the minimum; too small: learning is slow.
- By repeatedly applying these updates, the network gradually reduces the loss & improves its predictions.

(3)

a) ANN: Processes each input independently, no memory of past inputs.

RNN: Processes sequences by passing ~~hidden~~ hidden states from one step to the next, giving it short-term memory.

b) Gradient vanish & explode during backpropagation through many time steps. making it hard to learn long-range relationships.

c) Roles of gates in LSTMs:

- Input, forget & output gates control which information to add, remove & output from memory.
- Gates help preserve important information across long sequences.

d) By maintaining a cell state with controlled updates via gate, LSTMs allow gradients to flow over long sequences, preventing from vanishing

e) Example tasks:

ANN: Image classification

RNN: Short-sequence language modeling & sentiment analysis.

LSTM: Machine translation, speech recognition & long time series forecasting.

(4)

a) Ex of long-range dependency:

Sentence: The cat that chased mouse ran away.

- To understand that "ran away" refers to "the cat", the model must remember a word from earlier in the sentence.
- A standard RNN may struggle because it can forget earlier information over long sequences.

b) - The memory cell stores important context, and gates decide what to keep & discard, helping LSTMs remember key information over long sentences.

- In machine translation, if a phrase is no longer needed, the forget gate becomes active (close to 0) to clear old info & focus on the new part of the sentence.