Architecting Intelligence

Assignment 1

Name : Divyanshi Shukla
Roll No.: 240371

Answer 1:

(a) Regression is basically about finding a relationship between inputs and outputs.
Given some input features (like height, weight, hours studied, etc.), regression tries to find a formula that can predict the output value as accurately as possible.

In linear regression, we assume this relationship is linear. That means we are trying to fit a straight line (or a plane / hyperplane in higher dimensions) that best explains the data. The parameters beta decide how much each feature contributes to the output.
We measure how bad our predictions are using an error, which is the difference between the true value and the predicted value

We use squared error for three main reasons:

1. It penalizes large mistakes more
   Squaring makes big errors much worse than small ones, which is usually what we want.

2. It is mathematically convenient
   Squared error gives us a smooth, differentiable function, which makes optimization easier.

3. It has strong statistical meaning
   If we assume noise in data is Gaussian, minimizing squared error gives the most likely parameters.

So, minimizing squared error helps us find parameters that give predictions closest to the actual data in an overall sense.

#PragyaNotebook

DATE

PAGE

Ans 1) b) The cost function for linear regression is given by

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

Expanding :-

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

To find the optimal parameter β, we take the gradient of $J(\beta)$ w.r.t β

$$\nabla_\beta J(\beta) = \frac{1}{2} \nabla_\beta \left[(y - X\beta)^T (y - X\beta) \right]$$

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

$$\nabla_\beta J(\beta) = 0 \quad \text{(To minimize the cost)}$$

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

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

Assuming $X^T X$ is invertible $\therefore$ multiply both sides by $(X^T X)^{-1}$

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

Hence, this is the normal equation.

(c)

(c) Although the normal equation provides an exact solution, directly computing it is often impractical for large datasets. This is because it requires forming and inverting the matrix $X^T X$, whose size increases with the number of input features. As the number of features grows, this computation becomes slow and requires a ~~lo~~ large amount of money.

Also, if some features are highly correlated, the matrix $X^T X$ can become ill-conditioned or even non-invertible, leading to numerical instability. Due to these issues, iterative optimization methods such as gradient descent are commonly used in practice. These methods avoid matrix inversion, scale better to large datasets, and allow effecient learning even when the data size is very large.

Answer 2:

(a) Backpropagation is the algorithm used to train neural networks by adjusting weights and biases so that the prediction error becomes smaller.

The main idea is simple:

1. First, we do a forward pass to compute the output of the network.

2. Then, we compute how wrong the prediction is using a loss function.

3. After that, we propagate this error backwards through the network to find how much each weight and bias contributed to the error.

The reason this works efficiently is because of the chain rule from calculus.
A neural network is made of layers where each output depends on the previous layer. The chain rule allows us to break a complicated derivative into smaller, manageable derivatives, layer by layer.

This avoids recomputing gradients repeatedly and makes training deep networks practical.

(b) and (c)

(Ans2) b) Forward equations :

hidden neuron 1 :

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

Output neuron :

$$z_2 = w_2 a_1 + b_2$$
$$a_2 = \sigma(z_2) = \hat{y}$$

Loss function (Binary Cross - Entropy)

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

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

* Gradient w.r.t w_2

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

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

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

* Gradient w.r.t b_2

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

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

#PragyaNotebook

DATE
PAGE

$$\therefore \boxed{\frac{\partial L}{\partial b_2} = a_2 - y}$$

Gradient w.r.t w_1

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

$$\frac{\partial z_2}{\partial a_1} = w_2 \quad , \quad \frac{\partial a_1}{\partial z_1} = a_1(1-a_1), \quad \frac{\partial z_1}{w_1} = x$$

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

* Gradient w.r.t b_1

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

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

(c) In gradient descent, parameters are updated by moving them slightly in the opposite direction of the gradient, because that direction reduces the loss

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

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

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

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

Here η is the learning rate

- If the learning rate is too small, ~~too~~ training ~~becc~~ becomes very slow
- If it is too large, the updates may overshoot and the model may not converge
- So, the learning rate controls how big each update step is and plays a crucial role in stable and efficient learning.

Answer 3:

(a) An ANN processes inputs independently. Each input is treated as a fixed-size vector, and the network does not remember previous inputs. Because of this, ANNs are suitable for problems where data points do not depend on each other.

In contrast, an RNN is designed to process sequential data. It has a feedback connection that allows information from previous time steps to influence the current output. This makes RNNs suitable for time-series data, text, and speech, where order matters.

(b) Simple RNNs suffer from the vanishing gradient problem. When sequences are long, gradients propagated backward through many time steps become very small. As a result, the network fails to learn relationships between inputs that are far apart in time, making it difficult to capture long-term dependencies.

(c) LSTMs introduce a gating mechanism to control the flow of information. These gates decide:

- what information to keep,
- what to update,
- and what to discard.

The gates allow important information to pass through many time steps without being lost, helping the network remember relevant patterns over long sequences.

(d) LSTMs use a memory cell with nearly linear information flow. This structure allows gradients to pass through many time steps without shrinking significantly. As a result, LSTMs can learn long-term dependencies much more effectively than simple RNNs.

(e)

- **ANN**: House price prediction or image classification, where inputs are independent.

- **RNN**: Stock price prediction or speech recognition, where recent past information is important.

- **LSTM**: Machine translation or language modeling, where understanding long-term context is necessary.

Answer 4:

(a) Consider the sentence:

> "The book that the professor who joined last year recommended was very interesting."

To correctly use the verb "was", the model needs to remember that the main subject is "the book", not "professor". This dependency spans many words.

A standard RNN would struggle with this kind of long-range dependency because information from earlier words gradually fades as the sequence becomes longer. As a result, it may fail to maintain the correct grammatical relationship.

(b) An LSTM uses a memory cell that can store important information for a long time. The flow of information into and out of this cell is controlled by gates:

- The input gate decides what new information should be stored.
- The forget gate decides what information should be removed.
- The output gate decides what information should influence the current output.

This gated structure allows the LSTM to selectively remember relevant information and discard unnecessary details, making it effective for long sequences.

Machine Translation example:
When translating a long sentence, early words such as the gender or number of a noun may no longer be relevant later in the sentence. In such cases, the forget gate needs to be strongly active (close to 0) to remove outdated information so that the model can focus on more relevant context and produce an accurate translation.