Architecting Intelligence Krisha Gothi
240862

Assignment - 1

Q1 Derivation of the linear Regression Solution

a · Regression aims to model the relationship between input features $x_i \in \mathbb{R}^d$ & a continuous output $y_i \in \mathbb{R}$. In linear regression, we assume this relationship can be approximated by a linear function of the inputs. The parameter vector β defines a hyperplane in the feature space. Learning regression means finding the values of β such that the predicted outputs $\hat{y} = X\beta$ are as close as possible to the actual outputs y for all data points. ~~B~~ In essence, regression finds the best-fitting line through the data

Minimizing the squared error is meaningful because -

1) Penalizes large errors more strongly: Squaring the error gives higher weight to larger deviations, encouraging the model to avoid large prediction mistakes.

2) Ensures a smooth and convex objective:
The squared loss leads to a convex cost function $J(\beta)$, which has a single global minimum. This guarantees a unique and stable solution

3) Mathematical convenience:
The squared loss is differentiable, allowing us to compute gradients easily and derive a closed-form solution

4) Statistical justification:
If the noise in the target value is assumed to be independent and normally distributed with zero mean, minimizing squared error corresponds to maximum likelihood estimation of the parameters.

b least square cost function: $J(\beta) = \frac{1}{2}||y - X\beta||^2$

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

$$= \frac{1}{2}\left(y^Ty - \underbrace{y^TX\beta}_{scalar} - \beta^TX^Ty + \beta^TX^TX\beta\right)$$

$$(\because y^TX\beta = \beta^TX^Ty)$$

$$\therefore J(\beta) = \frac{1}{2}\left(y^Ty - 2\beta^TX^Ty + \beta^TX^TX\beta\right)$$

. differentiate w.r.t. β

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

Now, set $\frac{dJ(\beta)}{d\beta} = 0 \Rightarrow X^T X\beta - X^T y = 0$

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

$$\therefore \boxed{\beta = (X^T X)^{-1} X^T y}$$ (if $X^T X$ is invertible)

$$\therefore \boxed{X^T X\beta = X^T y} \rightarrow \text{normal equation}$$

C Inversion is problematic because of -

1. Computational cost :-
   inverting a $d \times d$ matrix cost $O(d^3)$
   for large d (ex - 10,000 features) this becomes impossible

2. Singularity :-
   If columns of X are highly correlated, $X^T X$ becomes non - invertible

3. Memory issues :-
   Storing $X^T X$ becomes infeasible for huge datasets

- Iterative methods
  1) Can be used in mini - batch or stochastic gradient descent
  2) Easily adapt to streaming or online data
  3) Are well - suited for distributed and parallel computing

Q2 Backpropagation in Neural Networks

a • Backpropagation is an efficient algorithm used to compute the gradients of the loss function w.r.t. all trainable parameters in a neural network. The main idea is to propagate the error from the output layer backward through the network, assigning credit or blame to each parameter based on how much it contributed to the final prediction error. These gradients are then used to update the parameters so that the loss is minimized

• Chain rule allows efficient computation of gradients in neural networks as gradients are computed in a layer-wise backward pass, avoiding redundant calculations. Also, the computational cost is proportional to a forward pass through the network. This makes training deep networks feasible even with many layers and parameters.

b Hidden neuron 1: $z_1 = w_1 x + b_1$ $a_1 = \sigma(z_1)$

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

binary cross-entropy loss:

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

using the chain rule.

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

$\left(\because \frac{\partial \mathcal{L}}{\partial z_2} = a_2 - y\right)$

$\because z_2 = w_2 a_1 + b_2$

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

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

Similarly,

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

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

again applying chain rule through z_2 and a_1:

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

$$\therefore \frac{\partial \mathcal{L}}{\partial w_1} = (a_2 - y)\, a_1\, w_2\, \underbrace{\sigma(z_1)\left(1 - \sigma(z_1)\right)}_{a_1(1-a_1)}\, x$$

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

Similarly,

$$\frac{\partial \mathcal{L}}{\partial b_1} = \frac{\partial \mathcal{L}}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1} \qquad \because \frac{\partial z_1}{\partial b_1} = 1$$

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

C — In gradient descent, the parameters of the network are updated iteratively in the direction opposite to the gradient of the loss function, in order to minimize the loss.

Let $\eta > 0$ be the learning rate

Parameter update equations

$$w_2 \leftarrow w_2 - \eta \frac{\partial \mathcal{L}}{\partial w_2} = w_2 - \eta (a_2 - y) a_1$$

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

$$w_1 \leftarrow w_1 - \eta \frac{\partial \mathcal{L}}{\partial w_1} = w_1 - \eta (a_2 - y) w_2 a_1 (1 - a_1) x.$$

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

- Role of Learning rate η

1) Learning rate controls the step size of each update.

2) If η is too small, training becomes very slow

3) If η is too large, updates may overshoot the minimum, causing divergence or oscillations.

4) A well-chosen learning rate ensures stable and efficient convergence of the loss

Q3 ANN vs RNN vs LSTMs

a An ANN processes inputs independently. Each input sample is treated as a fixed-size vector, and there is no memory of previous inputs, while an RNN processes ~~seq~~ sequences by taking one input at a time step and maintaing a hidden state that carries information from previous time steps.

b Simple RNNs struggle with long-term dependencies mainly due to the vanishing and exploding gradient problem during training.

c ~~s~~LSTM networks are designed to overcome the limitation of simple RNNs by using gating mechanisms that control the flow of information through the network. These gates help ~~to~~ decide what information to keep, update, or discard, allowing the model to preserve important information over long time intervals.

- Forgest gate - decides what info. to erase
- Input gate - decides what new info to store
- Output gate - controls what to expose

d
- Gates prevent vanishing gradients by enabling nearly constant error flow through the cell state
- Important info. can be retained for many time steps using the forget gate
- Unnecessary updates are avoided by the input gate
- Controlled exposure via output gate improves stability and learning

d The memory cell has constant error carousel:

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t$$

Gradient flows through C_t almost unchanged → prevents Vanishing

e 1) ANN - house price prediction
as each input (size, location, no. of rooms, etc) is fixed-length feature vector, and predictions do not depend on any sequence or order of past data

2) RNN - short-term time series forecasting
RNNs are suitable when recent past values influence the current prediction & long-term dependencies are limited

3) LSTM - language modeling or machine translation
LSTMs are ideal because they can capture long-range dependencies in sequences, such as grammatical structure & context across long sentences

Q4 LSTMs in Natural Language Processing

a Consider the sentence:
"The book that the professor who the students admired wrote was very popular."
To choose the correct verb "was", the model must remember that the main subject is "the book", not "professor" or "students".
The subject and verb are seperated by several intervening clauses,

creating a long-range dependency.

- A standard RNN would struggle to model this dependency because information about the subject ("book") must be preserved over many time steps. Due to the vanishing gradient problem, simple RNNs tends to forget earlier words in long sentences, making it difficult to correctly link distant words such as subjects and verbs.

b An LSTM has a memory cell that acts like a long-term notebook carrying information forward through the sequence. Unlike a standard RNN, this memory is updated in a controlled way using gates -

1) The input gate decides what new info is worth writing into the memory
2) The forget gate decides what old info. is no longer useful and should be erased
3) The output gate decides what part of the stored memory should be revealed at the current step.

Machine Translation example

Sentence - "He went to Paris. It is beautiful."

while translating "It", the model must

- Forget the earlier context "He went" → forget gate ≈ 0
- Keep the location "Paris" in memory

So the forget gate becomes active for irrelevant information