

ASSIGNMENT 2

Due Date : 29/12/25

THEORETICAL

Q1) Explain why a perceptron cannot solve XOR.

What changed when we introduced multi-layer perceptrons?

Q2) Why linear layers stacked on linear layers remain linear. Why gradients shrink in deep networks. Why ReLU solves vanishing gradients better than sigmoid.

Q3) Why positional encoding is necessary. Difference between sinusoidal PE and absolute PE
Why RoPE (rotary embeddings) helps with long contexts

Q4) What do Query, Key, and Value represent? Why do we scale attention scores by $\sqrt{d_k}$?
Why are diagonal values usually highest?

Q5) Why do we split attention into multiple heads? What d_{model} , h , and d_{head} represent?

Q6) Why greedy decoding is suboptimal. Why beam search yields better sequences. Provide an example where greedy decoding fails

SOLVING and CODING

Q1) If a transformer has:

$d_{model}=768$

$h=12$

Compute:

(a) d_{head}

(b) Total parameters for Q, K, V projection matrices for one layer

Q2) Given the attention scores from one row:

[2.0, 1.0, 0.0]

Compute the softmax values.

Q3) Write a PyTorch or NumPy function:

$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k})V$

Inputs:

- Q, K, V (matrices)

Outputs:

- Attention output
- Attention weights matrix

Q4) Modify Q3 so positions cannot attend to future tokens:

- Set those entries to `-inf`
- Apply softmax afterward

demonstrate that masking prevents access to future words.

Q5) Visualize Attention Heads. Feed a sentence into a pre trained transformer (e.g., BERT from HuggingFace), plot attention maps.

Q6) Try to complete this code implementation.

<https://colab.research.google.com/drive/1EdDgUAz9io88Cfod3avnyfbQHWJFh76q?usp=sharing>

Make a new colab notebook and use that for submission.