Assignment 2
Architecting Intelligence

Azaad Katiyar (240249)

December 29, 2025

1 Theoretical Questions

Q1. Explain why a perceptron cannot solve XOR. What changed when we introduced multi-layer perceptrons?

Answer:
A single perceptron cannot solve XOR because XOR is a non linear operation. A perceptron can only learn linear decision boundaries and XOR is not linearly separable.
Multi-layer perceptrons introduce hidden layers and non linear activation functions, which allow the network to learn non linear decision boundaries, and this makes XOR solvable.

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

Answer:
Stacking linear layers on top of each other results into another linear transformation.
Consider this : $y = W_2(W_1x) = (W_2W_1)x$
This still remains linear, it is just a matrix multiplication. While backpropagating, for the earlier layers, we repeatedly multiply small derivatives which shrinks the derivatives. This problem is called vanishing gradients.
ReLU solves the problem of vanishing gradients because the derivative of ReLU is 1 for positive inputs unlike the sigmoid function whose derivative is always less than 0.25. So, for the active neurons, since derivative is 1, there is no shrinking.

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

Answer:
Self-attention sees words all at once. It has no notion of order. Positional encoding is used to inject word order.
Sinusoidal Positional Encoding uses sine and cosine functions to represent positions which are encoded using mathematical waves. Since, values are fixed, no learning is required. It can work for longer sequences. On the other hand, in Absolute Positional Encoding, each position has its own fixed embedding vector which are learned during training. It cannot generalize well to longer sequences.
RoPE helps with long contexts because it encodes position via rotation, which preserves relative positional information. It can also work for long sequences.

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

Answer:
Queries ask questions, they contain what we ask. Keys provide matching info like document titles. And then values hold the document content.
We scale attention scores by $\sqrt{d_k}$ to prevent the dot products from becoming too large, which would push the softmax into saturation. This keeps gradients stable.
Diagonal values are the highest because a word is most similar to itself and thus, self-attention reinforces token identities.

Q5. Why do we split attention into multiple heads? What d_model, h, and d_head represent?

Answer:
If there is only one attention head, then there is only one viewpoint but multiple head attention allows the model to attend to information from different viewpoints like grammar, semantics, etc.
d_model : Embedding dimension
h : Number of heads
$d\_head : \frac{d\_model}{h}$

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

Answer:
Greedy decoding is suboptimal because at each time step, it picks the word with highest probability. It never reconsider past choices. This can lead to a globally inferior sequence because language is sequential.
Beam search is better because it maintains multiple candidate sequences simultaneously, explores multiple future paths and chooses the best overall sentence, not just best word.

Example where greedy decoding fails:
Consider a language model which is generating a sentence that begins with:
"The movie was ..."
At the next step, the model assigns the following probabilities:
$P(\text{"good"} \mid \text{"The movie was"}) = 0.55$
$P(\text{"not"} \mid \text{"The movie was"}) = 0.45$
Greedy decoding selected "good" because it has the highest immediate probability.

At the next step:
$P(\text{"at all"} \mid \text{"The movie was good"}) = 0.2$
So, the total probability of the greedy sequence is:
P("The movie was good at all") $= 0.55 \times 0.2 = 0.11$

However, if the model had selected "not" at the prior step:
$P(\text{"good"} \mid \text{"The movie was not"}) = 0.95$
Then, the total probability becomes:
P("The movie was not good") $= 0.45 \times 0.95 = 0.4275$

This shows that greedy decoding has failed because greedy decoding picks "good" due to higher local probability but the resulting sentence has low global probability and poor semantics. Beam search would keep both paths and correctly select "The movie was not good".

2 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

Solution:
(a) $d\_head = \frac{768}{12} = \mathbf{64}$
(b) Each Q, K, V : 768 x 768
So, Total = 3 x 768 x 768 = **1,769,472 parameters**

Q2) Given the attention scores from one row: $[2.0, 1.0, 0.0]$ Compute the softmax values.

Solution:
We know,

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum e^{x_i}}$$

$$\sum e^{x_i} = e^{2.0} + e^{1.0} + e^{0.0} = 7.389 + 2.718 + 1.0 = 11.107$$

$$\text{softmax}(2.0) = \frac{e^{2.0}}{\sum e^{x_i}} = \frac{7.389}{11.107} = 0.665$$

$$\text{softmax}(1.0) = \frac{e^{1.0}}{\sum e^{x_i}} = \frac{2.718}{11.107} = 0.245$$

$$\text{softmax}(0.0) = \frac{e^{0.0}}{\sum e^{x_i}} = \frac{1.0}{11.107} = 0.090$$

$$\textbf{Softmax} = [\mathbf{0.665}, \mathbf{0.245}, \mathbf{0.090}]$$