Generative AI Transformer

23B1031
Mrunal Vairagade

December 2024

Mentors: Aditya Neeraje and Yash Sabale
Week 2 and 3 are completed and week 4 is done to some extent

1 Forward Propagation

For layer l:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$
$$a^{(l)} = f(z^{(l)})$$

where:

- $W^{(l)}$: Weight matrix,
- $b^{(l)}$: Bias vector,
- $a^{(l-1)}$: Activation from previous layer,
- $f(\cdot)$: Activation function.

2 Backward Propagation

Gradients for weights and biases:

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \frac{\partial \mathcal{L}}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial W^{(l)}}$$
$$\frac{\partial \mathcal{L}}{\partial b^{(l)}} = \frac{\partial \mathcal{L}}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial b^{(l)}}$$
$$\frac{\partial \mathcal{L}}{\partial z^{(l)}} = \frac{\partial \mathcal{L}}{\partial a^{(l)}} \cdot f'(z^{(l)})$$

3 Activation Functions

3.1 Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

Derivative:

$$f'(x) = f(x)(1 - f(x))$$

3.2 ReLU

$$f(x) = \max(0, x)$$

Derivative:

$$f'(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$$

3.3 Leaky ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise.} \end{cases}$$

Derivative:

$$f'(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0.01 & \text{otherwise.} \end{cases}$$

3.4 Parametric ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha x & \text{otherwise.} \end{cases}$$

Derivative:

$$f'(x) = \begin{cases} 1 & \text{if } x > 0, \\ \alpha & \text{otherwise.} \end{cases}$$

3.5 Softmax

For classification:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Derivative:

$$\frac{\partial f(x_i)}{\partial x_j} = f(x_i)(\delta_{ij} - f(x_j))$$

where δ_{ij} is the Kronecker delta.

3.6 Hyperbolic Tangent (tanh)

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Derivative:

$$f'(x) = 1 - \tanh^2(x)$$

4 Loss Functions

4.1 Mean Squared Error (MSE)

For regression:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Gradient:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = \frac{2}{n}(\hat{y}_i - y_i)$$

4.2 Cross-Entropy Loss

For classification:

$$\mathcal{L} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

Gradient:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}$$

5 Attention Mechanism

5.1 Self-Attention

Given input embeddings X:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

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

Gradient of attention:

$$\begin{aligned}
\frac{\partial \text{Attention}}{\partial Q} &= \frac{\partial \text{softmax}}{\partial Q} \cdot V \\
\frac{\partial \text{Attention}}{\partial K} &= \frac{\partial \text{softmax}}{\partial K} \cdot V \\
\frac{\partial \text{Attention}}{\partial V} &= \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)
\end{aligned}$$

5.2 Multi-Head Attention

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O$$

$$\text{head}_i = \text{Attention}(QW_{Q_i}, KW_{K_i}, VW_{V_i})$$

6 Positional Encoding

Transformers use positional encodings to maintain sequence order:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

7 Layer Normalization

Layer normalization stabilizes training:

$$\text{LayerNorm}(x) = \gamma \cdot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

where:

- μ: Mean of x,
- σ^2: Variance of x,
- γ, β: Learnable parameters,
- ϵ: Small constant for numerical stability.

8 Residual Connections

Residual connections help gradient flow:

$$\text{Output} = \text{Layer}(x) + x$$

9 Feedforward Network in Transformers

Each transformer block contains a feedforward network:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

10 Gradient Descent Optimization

Weights are updated using gradient descent:

$$W^{(l)} = W^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial W^{(l)}}$$

$$b^{(l)} = b^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial b^{(l)}}$$

where η is the learning rate.

11 Conclusion

This report presents the mathematical foundations of Transformers, including forward and backward propagation, activation functions, loss functions, attention mechanisms, positional encoding, layer normalization, residual connections, and gradient descent optimization. These formulations are essential for understanding and implementing generative AI models.