Transformer Gradiant Calculation

November 28, 2024

Contents

1 Overview of the Forward Pass

The forward pass of the model involves the following steps:

1. **Token Embedding**: Converts input token indices to embeddings.
2. **Positional Encoding**: Adds positional information to the embeddings.
3. **Transformer Encoder Blocks**: Composed of multi-head attention and feed-forward networks with residual connections and layer normalization.
4. **Output Projection Layer**: Maps the final embeddings to logits over the vocabulary.
5. **Softmax Function**: Converts logits to probabilities.
6. **Loss Computation**: Calculates the cross-entropy loss between predicted probabilities and true labels.

2 Loss Function and Gradient Calculation

2.1 Cross-Entropy Loss

Given the predicted probabilities $\hat{y}_i$ and true labels y_i, the cross-entropy loss for a single sample is:

$$L = -\log(\hat{y}_{y_i}) \tag{1}$$

For a batch of N samples, the average loss is:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log(\hat{y}_{i,y_i}) \tag{2}$$

2.2 Gradient of Loss with Respect to Logits

The softmax function converts logits z_i to probabilities $\hat{y}_i$:

$$\hat{y}_{i,j} = \frac{e^{z_{i,j}}}{\sum_k e^{z_{i,k}}} \tag{3}$$

The gradient of the loss L with respect to the logits $z_{i,j}$ is:

$$\frac{\partial L}{\partial z_{i,j}} = \hat{y}_{i,j} - \delta_{j,y_i} \tag{4}$$

where δ_{j,y_i} is the Kronecker delta function:

$$\delta_{j,y_i} = \begin{cases} 1 & \text{if } j = y_i \\ 0 & \text{otherwise} \end{cases}$$

3 Backward Pass Through the Network

We will compute the gradients layer by layer, starting from the output and moving backward through the network.

3.1 Output Projection Layer

3.1.1 Forward Pass

The output projection layer computes the logits:

$$z = hW \tag{5}$$

where:

- $h \in \mathbb{R}^{N\times d}$: Input embeddings from the previous layer.
- $W \in \mathbb{R}^{d\times V}$: Weight matrix mapping to vocabulary size V.

3.1.2 Gradient Computations

Gradient with Respect to W

$$\frac{\partial L}{\partial W} = h^{\top} \left(\hat{Y} - Y \right) \tag{6}$$

where:

- $\hat{Y} \in \mathbb{R}^{N \times V}$: Predicted probabilities.
- $Y \in \mathbb{R}^{N \times V}$: One-hot encoded true labels.

Gradient with Respect to h

$$\frac{\partial L}{\partial h} = \left(\hat{Y} - Y \right) W^{\top} \tag{7}$$

3.2 Transformer Encoder Blocks

Each Transformer encoder block contains:

- Multi-head attention with residual connection and layer normalization.
- Feed-forward network with residual connection and layer normalization.

We will compute gradients for each component.

3.2.1 Multi-Head Attention

Forward Pass For each head h:

1. **Compute Queries, Keys, Values**:

$$Q_h = XW_Q^h, \quad K_h = XW_K^h, \quad V_h = XW_V^h \tag{8}$$

2. **Scaled Dot-Product Attention**:

$$S_h = \frac{Q_h K_h^{\top}}{\sqrt{d_k}} \tag{9}$$

3. **Apply Mask and Softmax**:

$$A_h = \text{softmax}(S_h + \text{mask}) \tag{10}$$

4. **Compute Attention Output**:

$$O_h = A_h V_h \tag{11}$$

5. **Output Projection**:

$$H_h = O_h W_O^h \tag{12}$$

6. **Aggregate Heads**:

$$H = \sum_{h=1}^{H} H_h \tag{13}$$

Gradient Computations **Gradient with Respect to H_h**
Since $H = \sum H_h$:

$$\frac{\partial L}{\partial H_h} = \frac{\partial L}{\partial H} \tag{14}$$

Gradient with Respect to W_O^h

$$\frac{\partial L}{\partial W_O^h} = O_h^\top \frac{\partial L}{\partial H_h} \tag{15}$$

Gradient with Respect to O_h

$$\frac{\partial L}{\partial O_h} = \frac{\partial L}{\partial H_h} {W_O^h}^\top \tag{16}$$

Gradient with Respect to A_h

$$\frac{\partial L}{\partial A_h} = \frac{\partial L}{\partial O_h} V_h^\top \tag{17}$$

Gradient with Respect to V_h

$$\frac{\partial L}{\partial V_h} = A_h^\top \frac{\partial L}{\partial O_h} \tag{18}$$

Gradient with Respect to Attention Scores S_h
Let $G_h = \frac{\partial L}{\partial A_h}$. Since $A_h = \text{softmax}(S_h)$, we have:

$$\frac{\partial L}{\partial S_h} = A_h \odot (G_h - (A_h \odot G_h)\mathbf{1}) \tag{19}$$

where:

- $\odot$: Element-wise multiplication.
- $\mathbf{1}$: Column vector of ones.

Gradient with Respect to Q_h and K_h

$$\frac{\partial L}{\partial Q_h} = \left(\frac{\partial L}{\partial S_h}\right) K_h \left(\frac{1}{\sqrt{d_k}}\right) \tag{20}$$

$$\frac{\partial L}{\partial K_h} = \left(\frac{\partial L}{\partial S_h}\right)^\top Q_h \left(\frac{1}{\sqrt{d_k}}\right) \tag{21}$$

Gradient with Respect to W_Q^h, W_K^h, W_V^h

$$\frac{\partial L}{\partial W_Q^h} = X^\top \frac{\partial L}{\partial Q_h} \tag{22}$$

$$\frac{\partial L}{\partial W_K^h} = X^\top \frac{\partial L}{\partial K_h} \tag{23}$$

$$\frac{\partial L}{\partial W_V^h} = X^\top \frac{\partial L}{\partial V_h} \tag{24}$$

Gradient with Respect to X
Accumulate contributions from Q_h, K_h, V_h:

$$\frac{\partial L}{\partial X} += \frac{\partial L}{\partial Q_h} {W_Q^h}^\top + \frac{\partial L}{\partial K_h} {W_K^h}^\top + \frac{\partial L}{\partial V_h} {W_V^h}^\top \tag{25}$$

3.2.2 Layer Normalization and Residual Connection

Forward Pass

1. **Residual Connection**:

$$X_{\text{residual}} = X + H \tag{26}$$

2. **Layer Normalization**:

$$X' = \text{LayerNorm}(X_{\text{residual}}) \tag{27}$$

Gradient Computations **Gradient with Respect to LayerNorm Output**

$$\frac{\partial L}{\partial X'} = \text{Gradient from Next Layer} \tag{28}$$

Compute Intermediate Variables

- Mean μ and variance σ^2:

$$\mu = \frac{1}{D}\sum_{i=1}^{D} X_{\text{residual},i}, \quad \sigma^2 = \frac{1}{D}\sum_{i=1}^{D} (X_{\text{residual},i} - \mu)^2 \tag{29}$$

- Normalized input $\hat{X}$:

$$\hat{X} = \frac{X_{\text{residual}} - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{30}$$

Gradient with Respect to Scale and Shift Parameters

$$\frac{\partial L}{\partial \gamma} = \sum_i \frac{\partial L}{\partial X'_i}\hat{X}_i \tag{31}$$

$$\frac{\partial L}{\partial \beta} = \sum_i \frac{\partial L}{\partial X'_i} \tag{32}$$

Gradient with Respect to Normalized Input

$$\frac{\partial L}{\partial \hat{X}} = \frac{\partial L}{\partial X'} \odot \gamma \tag{33}$$

Gradient with Respect to X_{residual}

$$\frac{\partial L}{\partial X_{\text{residual}}} = \frac{1}{\sqrt{\sigma^2 + \epsilon}} \left(\frac{\partial L}{\partial \hat{X}} - \frac{1}{D}\sum_i \frac{\partial L}{\partial \hat{X}_i} - \hat{X}_i \sum_i \left(\frac{\partial L}{\partial \hat{X}_i}\hat{X}_i \right) \right) \tag{34}$$

Gradient with Respect to X and H
Since $X_{\text{residual}} = X + H$:

$$\frac{\partial L}{\partial X} + = \frac{\partial L}{\partial X_{\text{residual}}} \tag{35}$$

$$\frac{\partial L}{\partial H} = \frac{\partial L}{\partial X_{\text{residual}}} \tag{36}$$

3.2.3 Feed-Forward Network

Forward Pass

1. **First Linear Layer**:

$$F = X'W_1 + b_1 \tag{37}$$

2. **ReLU Activation**:

$$A = \text{ReLU}(F) \tag{38}$$

3. **Second Linear Layer**:

$$G = AW_2 + b_2 \tag{39}$$

4. **Residual Connection and Layer Normalization**:

$$Y_{\text{residual}} = X' + G \tag{40}$$

$$Y = \text{LayerNorm}(Y_{\text{residual}}) \tag{41}$$

Gradient Computations **Gradient with Respect to Output Y**

Backpropagate through layer normalization and residual connection as previously described.

Gradient with Respect to G

$$\frac{\partial L}{\partial G} = \frac{\partial L}{\partial Y_{\text{residual}}} \tag{42}$$

Gradient with Respect to W_2 and A

$$\frac{\partial L}{\partial W_2} = A^\top \frac{\partial L}{\partial G} \tag{43}$$

$$\frac{\partial L}{\partial A} = \frac{\partial L}{\partial G} W_2^\top \tag{44}$$

Gradient Through ReLU Activation

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial A} \odot \mathbb{1}_{F>0} \tag{45}$$

where $\mathbb{1}_{F>0}$ is the indicator function:

$$\mathbb{1}_{F>0} = \begin{cases} 1 & \text{if } F > 0 \\ 0 & \text{otherwise} \end{cases}$$

Gradient with Respect to W_1 and X'

$$\frac{\partial L}{\partial W_1} = X'^\top \frac{\partial L}{\partial F} \tag{46}$$

$$\frac{\partial L}{\partial X'} += \frac{\partial L}{\partial F} W_1^\top \tag{47}$$

Accumulate Gradient from Residual Connection

$$\frac{\partial L}{\partial X'} += \frac{\partial L}{\partial Y_{\text{residual}}} \tag{48}$$

3.3 Positional Encoding

Positional encoding adds positional information to the embeddings:

$$E_{\text{pos}} = E + P \tag{49}$$

- If P is **fixed**, no gradients are computed for P, and gradients with respect to E pass through unchanged.
- If P is **learnable**, compute gradients with respect to P similarly to E.

3.4 Embedding Layer

3.4.1 Forward Pass

The embedding layer maps token indices I to embeddings E:

$$X = E[I] \tag{50}$$

3.4.2 Gradient Computations

For each token index i in the sequence:

$$\frac{\partial L}{\partial E_{I_i}} += \frac{\partial L}{\partial X_i} \tag{51}$$

We accumulate the gradient for each embedding corresponding to the token indices.

4 Parameter Updates

After computing all gradients, we update the parameters using gradient descent:

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta} \tag{52}$$

where:

- θ: Parameters (weights and biases) of the model.
- η: Learning rate.

For example:

- **Update Embedding Matrix**:

$$E[I_i] \leftarrow E[I_i] - \eta \frac{\partial L}{\partial E_{I_i}} \tag{53}$$

- **Update Weights in Linear Layers**:

$$W \leftarrow W - \eta \frac{\partial L}{\partial W} \tag{54}$$

5 Example Calculations

Consider a simple example with the following assumptions:

- Batch size $N = 1$.
- Vocabulary size V.
- Embedding dimension d.
- Input sequence length T.

5.1 Compute $\frac{\partial L}{\partial z}$

Given the predicted probabilities $\hat{y}$ and true label y:

$$\frac{\partial L}{\partial z_j} = \hat{y}_j - \delta_{j,y} \tag{55}$$

This vector has a non-zero component for each class.

5.2 Compute Gradients in Output Projection Layer

Weights W:

$$\frac{\partial L}{\partial W} = h^\top (\hat{y} - y_{\text{one-hot}}) \tag{56}$$

Input h:

$$\frac{\partial L}{\partial h} = (\hat{y} - y_{\text{one-hot}}) W^\top \tag{57}$$

5.3 Backpropagate Through Transformer Blocks

Repeat gradient computations for each block as outlined, ensuring to:

- Accumulate gradients at residual connections.
- Backpropagate through layer normalization carefully, considering mean and variance dependencies.

5.4 Update Parameters

For each parameter θ:

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta} \tag{58}$$

6 Key Takeaways

- **Chain Rule Application**: Gradients are computed by applying the chain rule backward through each layer.
- **Matrix Calculus**: Utilize matrix derivatives to compute gradients efficiently.
- **Residual Connections**: When layers have residual connections, gradients from both paths are added together.
- **Layer Normalization**: Requires careful computation due to dependencies between inputs (mean and variance).
- **Parameter Updates**: After computing gradients, parameters are updated using gradient descent or an optimizer.