Transformer with GLU Variant Pseudocode

Zheling Zhang

March 6, 2024

Algorithm 1 Transformer model with Gated Linear Unit (GLU) variant

Require: x, a sequence of token IDs.
Ensure: y, the output sequence after passing through the Transformer model with a GLU variant.

```
Hyperparameters: N, the number of layers; d_model, the dimensionality of token embeddings; d_ff, the dimensionality of the feedforward layer; h, the number of attention heads.
Parameters θ include all following parameters:
    W_e ∈ R^{d_vocab × d_model}, the token embedding matrix.
    W_p ∈ R^{max_position × d_model}, the positional embedding matrix.
    For each layer l ∈ [1...N]:
        W_Q^l, W_K^l, W_V^l ∈ R^{d_model × (d_model/h)}, attention parameter matrices for each head.
        W_O^l ∈ R^{(d_model/h) × d_model}, output projection matrix for multi-head attention.
        γ^l, β^l ∈ R^{d_model}, parameters for layer normalization before and after the multi-head attention, respectively.
        W_1^l ∈ R^{d_model × d_ff}, W_2^l ∈ R^{d_ff × d_model}, weights for the feedforward network.
        W_g^l ∈ R^{d_ff × d_model}, weights for the gating mechanism in the GLU variant.
        b_1^l, b_2^l, b_g^l ∈ R^{d_model}, biases for the feedforward network and the gating mechanism.
    W_u ∈ R^{d_model × d_vocab}, the unembedding matrix.
Initialize the output sequence y to an empty list.
Compute the embedded input sequence W_e^x = W_e[x] + W_p[pos], where pos is the position sequence.
for each layer l ∈ [1...N] do
    Apply layer normalization: X_norm = LayerNorm(E_x, γ^l, β^l).
    Calculate self-attention: Z = MultiHeadAttention(X_norm, W_Q^l, W_K^l, W_V^l, W_O^l).
    Apply residual connection and layer normalization: X_norm = LayerNorm(X_norm + Z, γ^l, β^l).
    Apply the first feedforward projection: F1 = W_1^l X_norm + b_1^l.
    Apply the GLU variant: G = GLU_Variant(F1, W_g^l, b_g^l).
    Apply the second feedforward projection: F2 = W_2^l G + b_2^l.
    Apply residual connection: E_x = X_norm + F2.
    Append the result to the output sequence y.
Compute the unembedded output: P = softmax(W_u y).
return P
function GLU_VARIANT(X, W_g, b_g)
    Split the input matrix into two equal parts: A, B = split(X, 2, axis = −1).
    Apply a non-linearity to the first part: A = tanh(A).
    Apply the gating mechanism to the second part: B = sigmoid(W_g B + b_g).
    Element-wise multiplication of the results: G = A × B.
    return G
```