DL-HW3

Jianing Zhang jz5212

Feb 28th 2024

1.1 Attention

Answer to (a)

Given queries $Q \in \mathbb{R}^{d \times n}$, keys $K \in \mathbb{R}^{d \times m}$, and values $V \in \mathbb{R}^{t \times m}$. Firstly, we compute the attention matrix

$$\mathrm{A} = softargmax_{\beta}(K^T Q)$$

, where $A \in \mathbb{R}^{m \times n}$.
Then, we find out the output H,

$$H = V \cdot A$$

, where $H \in \mathbb{R}^{t \times n}$

Answer to (b)

The scale factor β is used to adjust the range of attention scores before the softargmax operation. As the softargmax function can be sensitive to large values, leading to extreme probability distributions, this can cause the gradients to vanish during back-propagation.
Therefore, we need β to ensure the dot product of **Q** and **K** has a more moderate range.
The usual value of β is $\frac{1}{\sqrt{d_k}}$.

Answer to (c)

Given that β approaches positive infinity, indicating a decrease in temperature, the softargmax function's output will converge to a one-hot encoding. This effectively preserves a singular value vector v, characteristic of hard attention. In the context of fully connected layers, to maintain the integrity of the value vector without alteration, one may employ a linear transformation with an identity matrix as the weight and no bias term.

Answer to (d)

When the attention mechanism applies a weighted average to the value vectors, it can effectively diffuse the information across the output. The situation where the outputs represent a spread version of the value vectors occurs when the softmax function of the attention mechanism produces a uniform or nearly uniform distribution of weights.As β approaches zero, the attention weights become more uniform, meaning that every value vector contributes equally to the final output $\mathbf{h}$. And this is a soft attention. In fully connected architectures, attention dispersion can be enhanced by using bias-free linear layers with weights initialized or learned to ensure equal contribution from each input element, promoting uniform attention distribution and feature blending across the input space.

Answer to (e)

Let's assume the original key is k and it is perturbed by a small noise ϵ such that the new key is $\hat{k} = k + \epsilon$. After the perturbation, the output will be:

$$\hat{H} = \text{softmax}_\beta \left(\left[k_1^T Q, \cdots, (k_i + \epsilon)^T Q, \cdots, k_m^T Q \right] \right) V$$

With the perturbation ϵ being a zero-mean Gaussian with small variance, the effect on $\hat{H}$ can be described as follows:

$$\hat{H} \approx H + \frac{\partial}{\partial k_i} (\text{softmax}_\beta(K^T Q)) V \epsilon^T$$

But as the mean of ϵ is 0, a small perturbation to one of the k will not change H.

Answer to (f)

Given the original output H and the perturbed output $\hat{H}$, the change in output due to the perturbation can be approximated as:

$$\Delta H = \hat{H} - H$$

Applying the perturbation α to the ith key vector, we get:

$$\hat{H} = \text{softmax}_\beta \left(\left[k_1^T Q, \cdots, (\alpha k_i)^T Q, \cdots, k_m^T Q \right] \right) V$$

The change in the output can be written as:

$$\Delta H \approx \hat{H} - H = \text{softmax}_\beta \left(\left[k_1^T Q, \cdots, (\alpha k_i)^T Q, \cdots, k_m^T Q \right] \right) V - \text{softmax}_\beta \left(K^T Q \right) V$$

To first order, this can be approximated by the derivative of the softmax function with respect to the ith component, multiplied by the change in that component $(\alpha - 1) k_i^T Q$:

$$\Delta H \approx \frac{\partial \text{softmax}_\beta \left(K^T Q\right)}{\partial (k_i^T Q)} (\alpha - 1) k_i^T Q V$$

The actual change ΔH will depend on the exact values of Q, K, and V, as well as the factor α.

1.2 Multi-Headed Attention

(a)

Suppose we have h heads in total and for each head i, the queries, keys, and values are linearly projected into separate smaller dimensional spaces:

$$Q_i = W_Q^i Q, \quad K_i = W_K^i K, \quad V_i = W_V^i V$$

where $W_Q^i \in \mathbb{R}^{d \times \frac{d}{h}}$, $W_K^i \in \mathbb{R}^{d \times \frac{d}{h}}$, and $W_V^i \in \mathbb{R}^{t \times \frac{t}{h}}$ are the parameter matrices for the i-th head.

The attention output for each head i is calculated using the scaled dot-product attention mechanism:

$$H_i = \text{softmax}_\beta \left({K_i}^T Q_i\right) V_i$$

Therefore, the individual attention outputs for each head:

$$H_1 = \text{softmax}_\beta \left((K_1)^T Q_1\right) V_1$$

$$H_2 = \text{softmax}_\beta \left((K_2)^T Q_2\right) V_2$$

$$\vdots$$

$$H_h = \text{softmax}_\beta \left((K_h)^T Q_h\right) V_h$$

The final output H could be a weighted sum of the individual head outputs:

$$H = [H_1, H_2 \cdots H_h] W^0$$

(b)

Yes, it is similar as the CNN. Multi-headed attention in transformers is akin to using multiple kernels in CNNs. Both approaches employ multiple linear projections to parse different features from the input, and then combine these diverse feature representations into a unified output.

1.3 Self Attention

(a)

As we have h heads in total and for each head i, the queries, keys, and values are linearly projected into separate smaller dimensional spaces:

$$Q_i = CW_Q^i, \quad K_i = CW_K^i, \quad V_i = CW_V^i$$

The attention output for each head i is calculated using the scaled dot-product attention mechanism:

$$H_i = \text{softmax}_\beta \left({K_i}^T Q_i \right) V_i$$

Therefore, the individual attention outputs for each head:

$$H_1 = \text{softmax}_\beta \left((K_1)^T Q_1 \right) V_1$$

$$H_2 = \text{softmax}_\beta \left((K_2)^T Q_2 \right) V_2$$

$$\vdots$$

$$H_h = \text{softmax}_\beta \left((K_h)^T Q_h \right) V_h$$

The final output H could be a weighted sum of the individual head outputs:

$$H = [H_1, H_2 \cdots H_h] W^0$$

(b)

We need the potential encoder for self-attention when the input is sequential,such as language understanding where the order of words affects meaning. And if the sequence order is not relevant to the task at hand, we do not need it.

(c)

If we suppose that K is an orthogonal matrix and $Q = K$, then, $K^T Q = I$ so that it is like an identity layer.

(d)

If $\text{softargmax}_\beta(K^T Q)$ is a diagonal matrix, then the self-attention operation simply performs a linear projection at each position. And the proper name should be '1D convolution', as it involves applying a linear projection to each element individually.

0.1 (e)

Consider a self-attention mechanism where each position in the input sequence attends equally to itself and its immediate neighbors. Assume the input sequence has positional encodings added, and let's define the attention weights for position i with respect to positions $i-1$, i, and $i+1$:

$$\text{softargmax}_\beta(K^T Q) = A_{ij} = \begin{cases} \frac{1}{3}, & \text{for } j = i-1, i, i+1 \\ 0, & \text{otherwise} \end{cases}$$

The self-attention operation applies these weights across the sequence:

$$H = AV$$

Where V is the input sequence with positional encodings and A is the matrix of attention weights. When the self-attention weights A are structured in this way, the layer is analogous to a convolutional layer with a kernel size of 3, applying the same weight to each position and its immediate neighbors.

1.4 Transformer

(a)

Compared to the previous sequence-to-sequence models (such as RNNs and LSTMs), Transformers can process the entire input sequence in parallel. In previous models, neighboring inputs are inferred sequentially through the latent representations of preceding inputs, leading to a sequential processing method. However, the Transformer handles sequences in parallel and understands the relationships between elements through attention mechanisms, while maintaining positional information through positional encoding, which means it can effectively manage large input sequences.

(b)

Self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Firstly, self-attention decreases the computational workload per layer in contrast to recurrent layers. Secondly, self-attention allows for the parallelization of extensive computations. Lastly, self-attention empowers the network to more effectively learn long-range dependence.

(c)

Multi-head attention mechanism linearly project the queries, keys and values h times with different, learned linear projections to dk, dk and dv dimensions, respectively. And then, we concat all heads together and again projected to

have the final solution. The benefit is that it allows the model to learn from different representations simultaneously.

(d)

In this model, it use position-wise feed-forward networks. And iIn addition to attention sub-layers, each of the layers in the encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. It uses two linear transformations with a ReLU activation in between. And this allows the model to employ distinct parameters at the same position across various layers.

(e)

Layer normalization technique normalizes the activation of neurons within a layer. For each training example, the mean and standard deviation of the activation across the neurons are calculated independently.Each neuron's activations are normalized by subtracting the mean and dividing by the standard deviation, followed by the application of learned scale and shift parameters to optimize feature scaling and shifting. And in the Transformer architecture, it is applied after the residual connection in each encoder and decoder.

Question 1.5

(a)

The key difference is their approach to processing input images. While CNNs rely on convolutional layers to extract spatial features hierarchically across the image, ViT treats the image as a sequence of patches and processes them using self-attention mechanisms similar to those used in natural language processing tasks. In ViT, the action of partitioning images into patches can be likened to a convolutional layer with same kernel size and stride.

(b)

Firstly, the Transformer has both an encoder and a decoder, but the ViT only has an encoder. Secondly, the input of the Transformer consists of words, while the input of ViT consists of images. Thirdly, the output of the Transformer involves predicting the next token in a sequence or generating the output sequence, while the output of ViT is a linear layer followed by a softmax function for classification tasks, where the model predicts the class labels of the input images.

(c)

Positional embedding in ViT are optimized for processing image inputs and capturing spatial features by preserving locality. And it is learned in 1D embedding, while the transformers in a combination of sine and cosine functions.

(d)

ViT links a single hidden layer MLP to the output of multi-head attention as a classification head. In fine-tuning, the classification head is simplified to just a single-layer MLP.