10.6. Self-attention and Positional Encoding

Lecture based on “Dive into Deep Learning” **http://D2L.AI** (Zhang et al., 2020)

Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

Overview

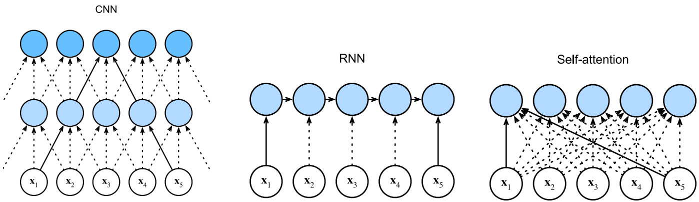
- CNNs or RNNs often encode sequences.
- Now we will feed a sequence of tokens into an attention mechanism such that each token has its own query, keys, and values.
- When computing the output for a token, it can attend via its query vector to each other token based on their keys.
- The output is a weighted sum over the other tokens.
- Because each token is attending to each other token, this architecture is called *self-attention*.
- Additional information for the sequence order can be added to each token.

Given a sequence of input tokens $\mathbf{x}_1, \ldots, \mathbf{x}_n$ where any $\mathbf{x}_i \in \mathbb{R}^d$ $(1 \leq i \leq n)$, its self-attention outputs a sequence of the same length $\mathbf{y}_1, \ldots, \mathbf{y}_n$, where

$$\mathbf{y}_i = f(\mathbf{x}_i, (\mathbf{x}_1, \mathbf{x}_1), \ldots, (\mathbf{x}_n, \mathbf{x}_n)) \in \mathbb{R}^d$$

according to the definition of attention pooling.
(batch size, number of time steps or sequence length in tokens, d)

Convolutional layer with kernel size k

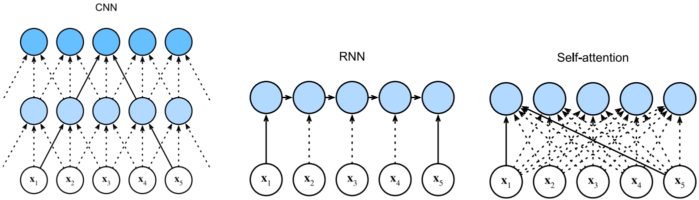


- For sequence length n, d input and output channels, the computational complexity of the convolutional layer is $\mathcal{O}(knd^2)$.
- CNNs are hierarchical, so there are $\mathcal{O}(1)$ sequential operations
- the maximum path length is $\mathcal{O}(log_k(n))$.

Example (two-layer CNN)

$\mathbf{x}_1$ and $\mathbf{x}_5$ are within the receptive field of a two-layer CNN with kernel size 3.

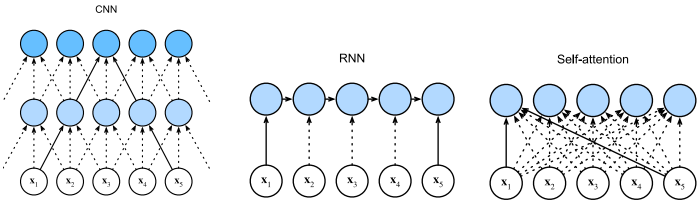
Recurrent Layer



- Updating the hidden state of RNNs involves multiplication of the $d \times d$ weight matrix and the d-dimensional hidden state.

  Computational complexity per update is $\mathcal{O}(d^2)$.
- For sequence length is n, the computational complexity of the recurrent layer is $\mathcal{O}(nd^2)$.
- There are $\mathcal{O}(n)$ sequential operations that cannot be parallelized
- the maximum path length is $\mathcal{O}(n)$.

Self-Attention



- Queries, keys, and values are $n \times d$ matrices.
- For the scaled dot-product, a $n \times d$ matrix is multiplied by a $d \times n$ matrix, then the output $n \times n$ matrix is multiplied by a $n \times d$ matrix.

  $\Rightarrow$ Self-attention has a $\mathcal{O}(n^2 d)$ computational complexity.
- Each token is directly connected to any other token via self-attention.
- Computation can be parallel with $\mathcal{O}(1)$ sequential operations
- The maximum path length is $\mathcal{O}(1)$.

- All in all, both CNNs and self-attention allow for parallel computation and self-attention has the shortest maximum path length.
- However, the quadratic computational complexity with respect to the sequence length makes self-attention prohibitively slow for very long sequences.

Why positional Encoding?

- Self-attention replaces sequential operations with parallel computation.
- However, self-attention by itself does not preserve the order of the sequence.
- *positional encodings* preserve information about the order of tokens as an additional input associated with each token.
- They can either be learned or fixed a priori.
- A simple scheme for fixed positional encodings is based on sine and cosine functions.

Positional encodings using trigonometric functions

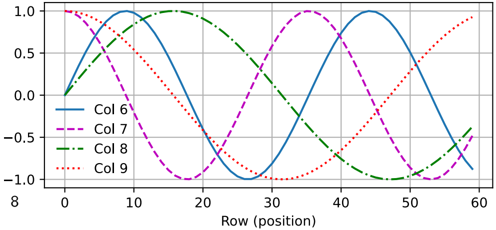
- $\mathbf{X} \in \mathbb{R}^{n \times d}$ d-dimensional inputs for n sequence tokens
- $\mathbf{P} \in \mathbb{R}^{n \times d}$ positional embedding matrix
- element on the i^{th} row and the $(2j)^{\mathrm{th}}$ column

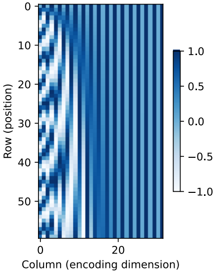
$$p_{i,2j} = \sin\left(\frac{i}{10000^{2j/d}}\right)$$

- element on the i^{th} row and the $(2j+1)^{\mathrm{th}}$ column

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

Positional encoding output
$\mathbf{X} + \mathbf{P}$





Summary

- In self-attention, the queries, keys, and values all come from the same representation.
- Both CNNs and self-attention enjoy parallel computation and self-attention has the shortest maximum path length.
- The quadratic computational complexity with respect to the sequence length makes self-attention prohibitively slow for very long sequences.
- To use the sequence order information, we can inject absolute or relative positional information by adding positional encoding to the input representations.