Deep Learning - Foundations and Concepts

Chapter 12. Transformers

nonlineark@github

March 17, 2025

Outline

Attention

The fundamental concept that underpins a transformer is attention:

- This was originally developed as an enhancement to RNNs for machine translation (Bahdanau, Cho and Bengio, 2014)
- Later, it was found that significantly improved performance could be obtained by eliminating the recurrence structure and instead focusing exclusively on the attention mechanism (Vaswani et al., 2017).

Attention

Consider the following two sentences:

- I swam across the river to get to the other bank.
- I walked across the road to get cash from the bank.

Here the word “bank” has different meanings in the two sentences:

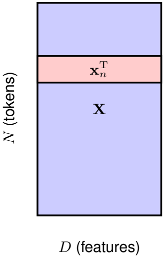
- In the first sentence, the words “swam” and “river” most strongly indicate that “bank” refers to the side of a river.
- In the second sentence, the word “cash” is a strong indicator that “bank” refers to a financial institution.

To determine the appropriate interpretation of “bank”, a neural network processing such a sentence should:

- Attend to specific words from the rest of the sequence.
- The particular locations that should receive more attention depend on the input sequence itself.

Transformer processing

- The input data to a transformer is a set of vectors $\{x_n\}$ of dimensionality D, where $n = 1, \ldots, N$.
- We refer to these data vectors as tokens, and the elements of the tokens are called features.
- We will combine the tokens into a matrix X of dimensions $N \times D$ in which the nth row comprises the token x_n^T.



Transformer processing

The fundamental building block of a transformer is a function that takes a data matrix X as input and creates a transformed matrix $\tilde{X}$ of the same dimensionality as the output:

$$\tilde{X} = \mathrm{TransformerLayer}(X)$$

A single transformer layer itself comprises two stages:

- The first stage, which implements the attention mechanism, mixes together the corresponding features from different tokens across the columns of the data matrix.
- The second stage acts on each row independently and transforms the features within each token.

Attention coefficients

Suppose we have a set of input tokens $x_1, \ldots, x_N$ and we want to map this set to another set $y_1, \ldots, y_N$:

- y_n should depend on all the tokens $x_1, \ldots, x_N$.
- This dependence should be stronger for those tokens x_m that are particularly important for determining the modified representation of y_n.

A simple way to achieve this is to define each output token y_n to be a linear combination of the input tokens:

$$y_n = \sum_{m=1}^{N} a_{nm} x_m$$

$$a_{nm} \geq 0 \qquad \sum_{m=1}^{N} a_{nm} = 1$$

Self-attention

The problem of determining the attention coefficients can be viewed from an information retrieval perspective:

- We could view the vector x_n as:
  - The key for input token n.
  - The value for input token n.
  - The query for output token n.
- To measure the similarity between the query x_n and the key x_m, we could use their dot product: $x_n^T x_m$.
- To make sure the attention coefficients define a partition of unity, we could use the $\mathrm{softmax}$ function to transform the dot products.

Self-attention

Dot-product self-attention:

$$y_n = \sum_{m=1}^{N} a_{nm} x_m$$

$$a_{nm} = \frac{\exp(x_n^T x_m)}{\sum_{m'=1}^{N} \exp(x_n^T x_{m'})}$$

Or write in matrix notation:

$$Y = \mathrm{softmax}(XX^T)X$$

where $\mathrm{softmax}(L)$ means to apply $\mathrm{softmax}$ to each row of the matrix L.