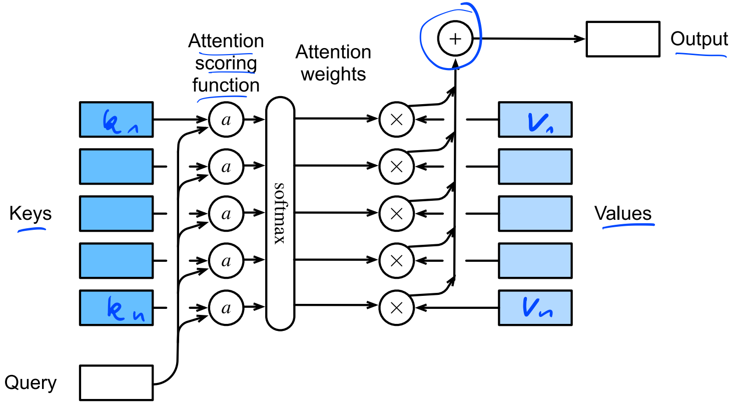
10.3. Attention Scoring functions and Multihead Attention

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

Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

Softmax attention



With the softmax operation to ensure nonnegative attention weights, *attention scoring functions* a are simpler to compute.

Many distance-based kernels to model interactions between queries and keys are more expensive to compute than inner products.

Scaled dot-product attention function

Let's review the attention function (without exponentiation) from the Gaussian kernel:

$$\begin{aligned} \exp\left(a(\mathbf{q}, \mathbf{k}_i)\right) &= \overset{\exp}{\left(-\frac{1}{2}\|\underline{\mathbf{q}} - \underline{\mathbf{k}_i}\|^2\right)} \\ &= \underline{\underline{\mathbf{q}^\top \mathbf{k}_i}} \boxed{- \frac{1}{2}\underline{\|\mathbf{k}_i\|^2} - \frac{1}{2}\underline{\underline{\|\mathbf{q}\|^2}}.} \end{aligned}$$

- $\frac{1}{2}\|\mathbf{q}\|^2$ is identical for all $(\mathbf{q}, \mathbf{k}_i)$ pairs.
  $\Rightarrow$ After normalizing attention weights to 1, this term disappears.
- Batch and layer normalization lead to activations that have bounded (often constant) norms $\|\mathbf{k}_i\| \approx \mathrm{const}$.
  $\Rightarrow$ $\|\mathbf{k}_i\|$ can also be removed from the definition of a.

After re-scaling by $1/\sqrt{d}$ we get a commonly used *scaled dot-product attention* attention function:

$$a(\mathbf{q}, \mathbf{k}_i) = \underline{\mathbf{q}^\top \mathbf{k}_i}/\sqrt{d}.$$

Assume that all the elements of the query $\mathbf{q} \in \mathbb{R}^d$ and the key $\mathbf{k}_i \in \mathbb{R}^d$ are independent and identically drawn random variables with zero mean and unit variance.
The dot product between both vectors has zero mean and a variance of d.
Re-scaling by $1/\sqrt{d}$ ensures that the variance of the dot product remains one regardless of vector length.

We can simplify this further by using the softmax operation:

$$\underline{\alpha(}\mathbf{q}, \mathbf{k}_i) = \underline{\mathrm{softmax}(}a(\mathbf{q}, \mathbf{k}_i)) = \frac{\exp(\underline{\mathbf{q}^\top \mathbf{k}_i}/\sqrt{d})}{\sum_{j=1} \exp(\mathbf{q}^\top \mathbf{k}_j/\sqrt{d})}.$$

Masked Softmax Operation

For sequence models, we may have sequences of different lengths.
E.g., sequences with different length may be in the same minibatch, necessitating padding with dummy tokens for shorter sequences.

Example (Padded text sequences with different length)

Dive into Deep Learning //
Learn to code <blank> //
Hello world <blank> <blank> // $l = 2$

Since we do not want <blank> in our attention model we limit $\sum_{i=1}^{n} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i$ to $\sum_{i=1}^{l} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i$, where $l \leq n$ is the actual sentence length.

Batch Matrix Multiplication

$$\mathbf{Q} = [\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_n] \in \mathbb{R}^{n \times a \times b} \mathbf{K} = [\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_n] \in \mathbb{R}^{n \times b \times c}$$

Batch matrix multiplication (BMM) computes the element-wise product

$$\mathrm{BMM}(\mathbf{Q}, \mathbf{K}) = [\mathbf{Q}_1\mathbf{K}_1, \mathbf{Q}_2\mathbf{K}_2, \ldots, \mathbf{Q}_n\mathbf{K}_n] \in \mathbb{R}^{n \times a \times c}.$$

E.g., scaled dot-product attention using BMM:

- n Queries $\mathbf{Q} \in \mathbb{R}^{n \times d}$ of length d
- m Keys $\mathbf{K} \in \mathbb{R}^{m \times d}$ of length d
- m Values $\mathbf{V} \in \mathbb{R}^{m \times v}$ of length v

$$\mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \in \mathbb{R}^{n \times v}.$$

Another commonly used operation is to multiply batches of matrices with another. When we have minibatches of queries, keys, and values we can multiply batches of matrices.

additive attention scoring

When queries $\mathbf{q}$ and keys $\mathbf{k}$ are vectors of different dimensionalities, we can either use a matrix to address the mismatch via $\mathbf{q}^\top \mathbf{M}\mathbf{k}$, or we can use additive attention as the scoring function.
Given a query $\mathbf{q} \in \mathbb{R}^q$ and a key $\mathbf{k} \in \mathbb{R}^k$, the *additive attention* scoring function is given by

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^\top \text{tanh}(\mathbf{W}_q\mathbf{q} + \mathbf{W}_k\mathbf{k}) \in \mathbb{R},$$

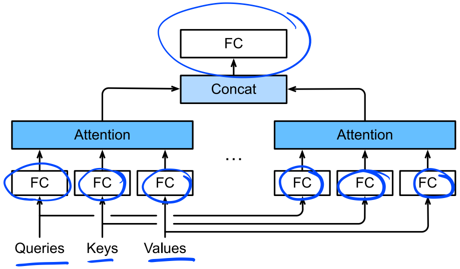
where $\mathbf{W}_q \in \mathbb{R}^{h\times q}$, $\mathbf{W}_k \in \mathbb{R}^{h\times k}$, and $\mathbf{w}_v \in \mathbb{R}^h$ are the learnable parameters.

Softmax ensures nonnegativity and normalization.

$$\operatorname{softmax}\left(\mathbf{w}_v^\top \text{tanh}(\mathbf{W}_q\mathbf{q} + \mathbf{W}_k\mathbf{k})\right).$$

An equivalent interpretation of is that the query and key are concatenated and fed into an MLP with a single hidden layer.

Multi-Head Attention



- Queries, keys, and values are projected with h different linear layers.
- h projected queries, keys, and values are fed into attention pooling in parallel.
- h attention pooling outputs are concatenated and transformed with another learned linear projection to produce the final output.
- Each of h attention pooling outputs is a *head*.

Given the same set of queries, keys, and values we may want our model to combine different behaviors of the same attention mechanism, such as capturing dependencies of various ranges (e.g., shorter-range vs. longer-range) within a sequence.
Allow attention to jointly use different representation subspaces of queries, keys, and values by transforming with h independently learned linear projections.

Mathematical Definition

Given a query $\mathbf{q} \in \mathbb{R}^{d_q}$, a key $\mathbf{k} \in \mathbb{R}^{d_k}$, and a value $\mathbf{v} \in \mathbb{R}^{d_v}$, each attention head $\mathbf{h}_i$ $(i = 1, \ldots, h)$ is computed as

$$\mathbf{h}_i = f(\mathbf{W}_i^{(q)}\mathbf{q}, \mathbf{W}_i^{(k)}\mathbf{k}, \mathbf{W}_i^{(v)}\mathbf{v}) \in \mathbb{R}^{p_v},$$

where learnable parameters $\mathbf{W}_i^{(q)} \in \mathbb{R}^{p_q \times d_q}$, $\mathbf{W}_i^{(k)} \in \mathbb{R}^{p_k \times d_k}$ and $\mathbf{W}_i^{(v)} \in \mathbb{R}^{p_v \times d_v}$, and f is attention pooling, such as additive attention and scaled dot-product attention.

The multi-head attention output is a linear transformation of the concatenation of h heads:

$$\mathbf{W}_o \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_h \end{bmatrix} \in \mathbb{R}^{p_o},$$

where $\mathbf{W}_o \in \mathbb{R}^{p_o \times h p_v}$ are learnable parameters.

Based on this design, each head may attend to different parts of the input.
More sophisticated functions than the simple weighted average can be expressed.

Summary

- In this section we introduced the two key attention scoring functions:
    - dot product
    - additive attention
- They can aggregating across sequences of variable length.
- Dot product attention is the mainstay of modern Transformer architectures.
- When queries and keys are vectors of different lengths, we can use the additive attention scoring function instead.
- Optimizing these layers has been a key area of advance recently.
    - Nvidia's Transformer Library
    - Megatron (NVIDIA) for Large Language Models
- Multi-head attention combines knowledge of the same attention pooling via different representation subspaces of queries, keys, and values.