Data-dependent weights

Consider a vanilla MLP layer:

$$y = \sigma(Wx + b)$$

Writing out the linear transformation:

$$y_i = \sum_{j=1}^{n} \underbrace{w_{ij}}_{\text{learned weights}} x_j$$

with w_{ij} being learned, fixed (data-independent) weights.

Now, say we want the weights to depend on the whole input x; i.e, the coefficient that multiplies x_j should depend on the element being transformed,

$$y_i = \sum_{j=1}^{n} \underbrace{\alpha_{ij}}_{\text{input-dependent weights}} x_j \; ; \quad \alpha_{ij} = f(x_i, x_j)$$

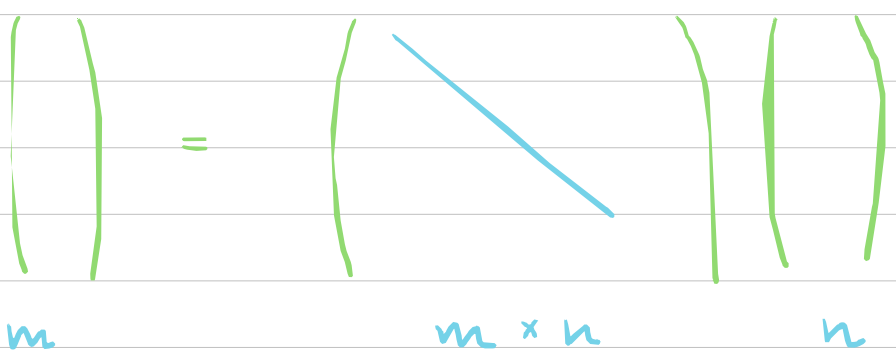
The simplest choice is a product. $x_i x_j$ ← similarity s_{ij}

Normalize via softmax,

$$\alpha_{ij} = \frac{e^{x_i x_j}}{\sum_{k=1}^{n} e^{x_i x_k}} \; ; \quad \alpha_{ij} > 0, \; \sum_j \alpha_{ij} = 1$$

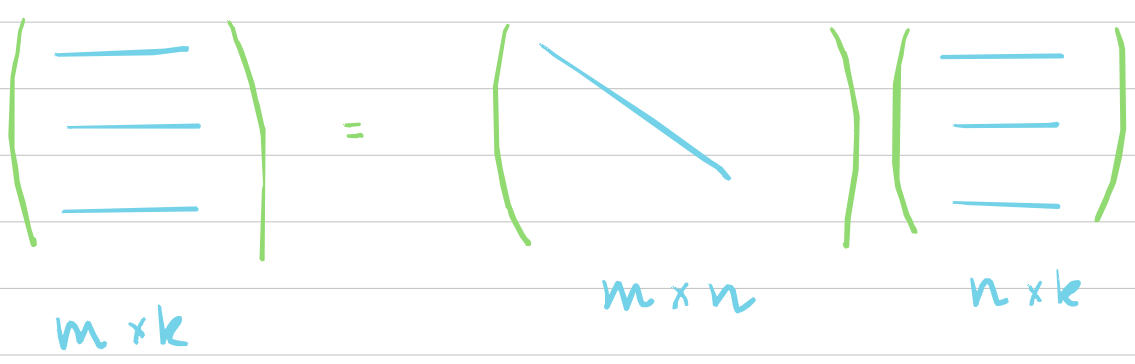
—

We can generalize this (with some notation abuse) to "vector" $\vec{x}_i, \vec{y}_i$. So previously we had



m $\quad$ $m \times n$ $\quad$ n

whereas now we have



$m \times k$ $\quad$ $m \times n$ $\quad$ $n \times k$

Everything before works, except $x_i x_j \rightarrow x_i^T x_j$.

Additionally, since the dot product commutes, we can learn representations of unordered sets of arbitrary cardinality

—

So far, nothing is learned. Simple learned similarity S_{ij} - project the inputs via learned projection

$$S_{ij} = (W_s x_i)^T (W_s x_j) = x_i^T \underbrace{(W_s^T W_s)}_{\text{Learned similarity metric}} x_j$$

Attention

Go a step further - different projections for x_i and x_j when computing similarity, and also project what is being summed over:

$$S_{ij} = \underbrace{(W_Q x_i)^T}_{\text{what elem. } i \text{ asks}} \underbrace{(W_K x_j)}_{\text{what elem. } j \text{ returns}}$$

$$y_i = \sum_j \underbrace{\text{softmax}(S_{ij})}_{\alpha_{ij} \text{ - attention weights}} \underbrace{W_V x_j}_{\text{Contribution of elem. } j}$$

$\in \mathbb{R}^{d_v}$

—

$$q_i = W_Q x_i \quad \in \mathbb{R}^{d_k} \quad \text{Query}$$

$$k_j = W_K x_j \quad \in \mathbb{R}^{d_k} \quad \text{Key}$$

$$v_j = W_V x_j \quad \in \mathbb{R}^{d_v} \quad \text{Value}$$

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$\in \mathbb{R}^{n \times n}$ ← set cardinality

$\sqrt{d_k}$ - normalization to account for scaling of dot product with dimension

Attention is permutation equivariant - swapping order of inputs correspondingly swaps the output order.

x $\quad$ y

This is scaled dot-product attention