Attention is all you need

VLC = ∭ Vision Learning Control dVdLdC

A little attention, please?

Jonathon Hare

Vision, Learning and Control
University of Southampton

Core idea: Attending to part of a vector or tensor

Static attention

$$\hat{\mathbf{X}} = \text{softmax}(\mathbf{W})\mathbf{X}$$

or, *factorised*,

$$\hat{\mathbf{X}} = \text{softmax}(\mathbf{W}_1\mathbf{W}_2)\mathbf{X}$$

Dynamic Attention

$$\hat{\mathbf{X}} = f(\mathbf{Z}, \theta)\mathbf{X}$$

or, *factorised*,

$$\hat{\mathbf{X}} = f(\mathbf{Z}_f, \theta_f)g(\mathbf{Z}_g, \theta_g)\mathbf{X}$$

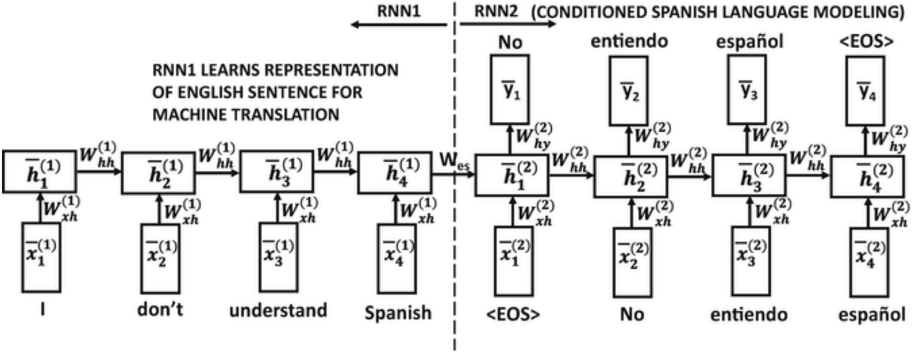
(Dynamic) Attention vs Self-attention

- In regular attention, the weights applied to $\mathbf{X}$ are computed using some additional auxiliary input (e.g. $\mathbf{Z}$)

(Dynamic) Attention vs Self-attention

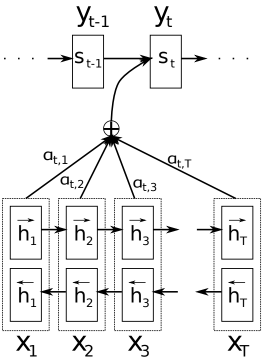
- In regular attention, the weights applied to $\mathbf{X}$ are computed using some additional auxiliary input (e.g. $\mathbf{Z}$)
- Self-attention is only computed as a function of $\mathbf{X}$ (equivalently $\mathbf{Z} = \mathbf{X}$)

Dynamic Attention Example - Seq2Seq models



https://link.springer.com/chapter/10.1007/978-3-319-73531-3_10

Dynamic Attention Example - Seq2Seq models



$$\alpha_t = \text{softmax}([\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_1), \dots, \text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_T)]^\top)$$

$$\text{score}(\boldsymbol{s}, \boldsymbol{h}) = \boldsymbol{v}^\top \tanh\left(\boldsymbol{W}[\boldsymbol{s}; \boldsymbol{h}]\right)$$

$$c = \alpha_t^\top \boldsymbol{H} \text{ where } \boldsymbol{H} = [h_1, h_2, \dots, h_T]^T$$

commonly known as “Additive Attention”, even though its based on concatenation!

Bahdanau, D., Cho, K. and Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. ICLR 2015.

Hard Attention vs Soft-attention

- Soft-attention: use the softmax to smoothly attend mostly to one thing (but capture a bit of everything else)
- Hard attention: you specifically only attend to one thing: tricks (e.g. policy gradients or ST operator) from last lecture required to learn

Aside: Relaxation of a map/hashtable/dictionary

Scaled dot-product attention

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}})\boldsymbol{V}$$

- In the previous Seq2Seq example we could replace additive attention with scaled dot-product attention with something like $\boldsymbol{Q} = f(\boldsymbol{s}_{t-1})$, $\boldsymbol{K} = g(\boldsymbol{H})$ and $\boldsymbol{V} = j(\boldsymbol{H})$.

Scaled dot-product attention

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}})\boldsymbol{V}$$

- In the previous Seq2Seq example we could replace additive attention with scaled dot-product attention with something like $\boldsymbol{Q} = f(\boldsymbol{s}_{t-1})$, $\boldsymbol{K} = g(\boldsymbol{H})$ and $\boldsymbol{V} = j(\boldsymbol{H})$.
- The scaling $1/\sqrt{d_k}$ is just to improve learning (larger d_k implies larger dot products, which pushes further towards the flatter bit of the softmax, and thus smaller gradients.)

Scaled dot-product self-attention

$$\text{SelfAttention}(\boldsymbol{X}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}})\boldsymbol{V}$$

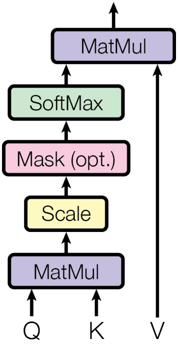
$$\boldsymbol{Q} = \boldsymbol{W}_q\boldsymbol{X}$$

$$\boldsymbol{K} = \boldsymbol{W}_k\boldsymbol{X}$$

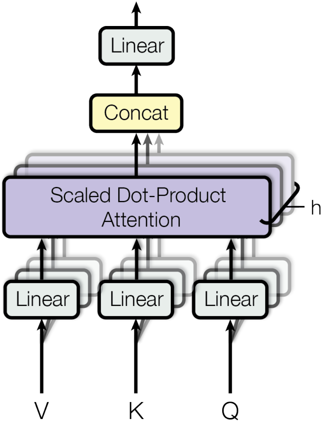
$$\boldsymbol{V} = \boldsymbol{W}_v\boldsymbol{X}$$

Multi-head Attention

Scaled Dot-Product Attention



Multi-Head Attention



$$\mathrm{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = [\mathrm{head}_1; \ldots; \mathrm{head}_n]\boldsymbol{W}^O$$

$$\mathrm{head}_i = \mathit{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

The Transformer