# Transformers - Mathematical derivation.

From the paper *Attention is all you need, Vaswani et al. 2017*

Input: $X_i \in \mathbb{R}^d$. $(X_i)_{1\leq i\leq n}$.

## ① Multi-head attention

Attention vectors are computed from each input $X_i$, $1\leq i\leq n$, and independently for each "head" $h$, $1\leq h\leq H$.

Keys : $k_h(x_i) = W_{h,k}^T X_i$

Queries : $q_h(x_i) = W_{h,q}^T X_i$

Values : $v_h(x_i) = W_{h,v}^T x_i$

## ② Attention weights

For all $1\leq i,j\leq n$, $1\leq h\leq H$,

$$\alpha_h(i,j) = \mathrm{Softmax}\left(\{q_h(x_i)^T k_h(x_j)\}_{1\leq j\leq n}\right)$$

where $\mathrm{Softmax}(z_1,\dots,z_n) = \frac{1}{\sum_{1\leq i\leq n} e^{z_i}}\left(e^{z_1},\dots,e^{z_n}\right)$.

## ③ Mixture of Values

. Define for all $1\leq i\leq n$

$$u_i = \sum_{h=1}^{H} W_{u,h}^T \Big( \underbrace{\sum_{j=1}^{n} \alpha_h(i,j) v_h(x_j)}_{\text{Mixture of values for each input on head } h.} \Big)$$

. Layer normalization of the $(u_i)$. → $u_i \leftarrow \mathrm{Layer\ norm}(u_i + x_i)$.

## ④ outputs.

. For all $1\leq i\leq n$ $\quad z_i = W_{z,1}^T\, \sigma\left(W_{z,2}^T u_i\right)$

. Layer normalization of the $(z_i)$. → $z_i \leftarrow \mathrm{Layer\ norm}(z_i + u_i)$.

Layer normalization of a vector $(v_1,\dots,v_n) = v$: $\quad v \leftarrow \beta_1 \underset{\text{empirical std}}{\sigma_v^{-1}} (v - \underset{\text{empirical mean}}{\mu_v}) + \beta_2$

Steps 1 to 4 provide a Regression function $T_\theta : (x_1,\dots,x_n) \longmapsto (z_1,\dots,z_n)$.

In practice, a Transformer network is given by: $T_{\theta_L} \circ \dots \circ T_{\theta_1}$ !

## ⑤ Positional encoding.

Inputs are considered as unordered vectors to compute and assign attention weights. If input data are sequential (i.e. $i$ refers to a time index), several additional positional encodings have been considered.

one-hot : $X_i \leftarrow (X_i, e_i)^T$ $e_i$, i-th canonical vector of $\mathbb{R}^n$.

Sinusoidal: $p_{k,2i} = \sin\left(\frac{k}{\gamma^{2i/d}}\right)$ ; $p_{k,2i+1} = \cos\left(\frac{k}{\gamma^{2i/d}}\right)$

$$z = W^T \sigma(W^T u) + p.$$

(6) Connection to RNN - Time series.

Next session!

# Transformers for time series - Similarities with "LSTM"

From the paper LSTM as a dynamically computed element-wise weighted sum, Lévy et al. (2018).

Long short term memory (1997, LSTM) are very popular networks to perform prediction for time series.

In this case, the data $(X_t)_{t \geq 0}$ are sequential and $t$ stands for time.

The prediction of a new data is based on intermediate representation $\{(c_t, h_t)\}_{t \geq 0}$ computed recursively:

$$\tilde{c}_t = \sigma(W_1 h_{t-1} + W_2 X_t) \quad \| \quad \text{Content layer}$$

$$\left.\begin{aligned} i_t &= \sigma(W_3 h_{t-1} + W_4 X_t) \\ f_t &= \sigma(W_5 h_{t-1} + W_6 X_t) \\ (*) \quad c_t &= i_t \tilde{c}_t + f_t c_{t-1} \end{aligned}\right\| \quad \text{Memory Layer}$$

$$\left.\begin{aligned} o_t &= \sigma(W_7 h_{t-1} + W_8 X_t) \\ h_t &= o_t \sigma(c_t) \end{aligned}\right\| \quad \text{output Layer}$$

Recursive formulation of (*): $c_t = \sum_{j=0}^{t} i_j \left(\prod_{k=j+1}^{t} f_k\right) \tilde{c}_j$

Proof: Assume that the result holds at $t$.

$$\begin{aligned} c_{t+1} &= i_{t+1} \tilde{c}_{t+1} + f_{t+1} c_t \\ &= i_{t+1} \tilde{c}_{t+1} + f_{t+1} \sum_{j=0}^{t} i_j \left(\prod_{k=j+1}^{t} f_k\right) \tilde{c}_j \\ &= \underline{\sum_{j=0}^{t+1} i_j \left(\prod_{k=j+1}^{t+1} f_k\right) \tilde{c}_j} \end{aligned}$$

of the form $\sum_{j=0}^{t+1} \omega_j^{t+1} \tilde{c}_j$

In Lévy et al., authors simplified a bit the LSTM to understand the element-wise weighted sum.

Assume to simplify that
$$\begin{cases} \tilde{c}_t = \sigma(W_1 x_t) \\ i_t = \sigma(W_2 x_t) \\ f_t = \sigma(W_3 x_t) \end{cases}$$

Then, $c_t = \sum_{j=0}^{t} \underbrace{i_j}_{\sigma(W_2 x_j)} \left(\prod_{k=j+1}^{t} \overbrace{f_k}^{\sigma(W_3 x_k)}\right) \underbrace{\tilde{c}_j}_{\sigma(W_1 x_j)}$ : Element-wise weighted sum of featurized inputs.

Reminder for the Transformers: $u_i = \sum_{h=1}^{H} W_{u,h}^T \left(\sum_{j=1}^{n} \alpha_h(i,j) v_h(x_j)\right)$

$$\underset{\substack{H=1 \\ W_{u,h}=Id}}{=} \sum_{j=1}^{n} \alpha_h(i,j) v_h(x_j) \underset{\text{Mask future values.}}{=} \sum_{j=1}^{t} \tilde{\alpha}(i,j) v_h(x_j)$$

Important difference: In LSTM, $\|f_k\| < 1$ so that attention weights decrease (fast) in the past!