Transformer

Seq2seq model with “Self-attention”
Sequence

Previous layer

Next layer

Hard to parallel!

Using CNN to replace RNN
Sequence

Previous layer
- $a^1$
- $a^2$
- $a^3$
- $a^4$

Next layer
- $b^1$
- $b^2$
- $b^3$
- $b^4$

Hard to parallel

Filters in higher layer can consider longer sequence

Using CNN to replace RNN (CNN can parallel)
Self-Attention

$\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$ can be parallelly computed.

$b^i$ is obtained based on the whole input sequence.

$\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4$ can be parallelly computed.

You can try to replace any thing that has been done by RNN with self-attention.
**Self-attention**

https://arxiv.org/abs/1706.03762

Attention is all you need.

$q$: query (to match others)

\[ q^i = W^q a^i \]

$k$: key (to be matched)

\[ k^i = W^k a^i \]

$\nu$: information to be extracted

\[ \nu^i = W^\nu a^i \]

\[ a^i = W x^i \]
**Self-attention**

拿每個 query q 去對每個 key k 做 attention

Scaled Dot-Product Attention:  \[ \alpha_{1,i} = \frac{q^1 \cdot k^i}{\sqrt{d}} \]

d is the dim of q and k

dot product
Self-attention

\[ \hat{\alpha}_{1,i} = \exp(\alpha_{1,i}) / \sum_j \exp(\alpha_{1,j}) \]

\[ \alpha_{1,1}, \alpha_{1,2}, \alpha_{1,3}, \alpha_{1,4} \]

\[ q^1, k^1, v^1 \]
\[ q^2, k^2, v^2 \]
\[ q^3, k^3, v^3 \]
\[ q^4, k^4, v^4 \]

\[ a^1, a^2, a^3, a^4 \]
\[ x^1, x^2, x^3, x^4 \]
Self-attention

Considering the whole sequence

\[ b^1 = \sum_i \hat{\alpha}_{1,i} v^i \]
**Self-attention**

拿每個 query q 去對每個 key k 做 attention

$$b^2 = \sum_i \alpha_{2,i} v^i$$
Self-attention

\( b^1, b^2, b^3, b^4 \) can be parallelly computed.
**Self-attention**

$$q^i = W^q a^i$$

$$k^i = W^k a^i$$

$$v^i = W^v a^i$$

$$Q = W^q a^1 a^2 a^3 a^4$$

$$K = W^k a^1 a^2 a^3 a^4$$

$$V = W^v a^1 a^2 a^3 a^4$$

$$x^i$$
Self-attention

\[ b^1 \]

\( \hat{\alpha}_{1,1} \)
\( q^1 \) \( k^1 \) \( v^1 \)
\( \hat{\alpha}_{1,2} \)
\( q^2 \) \( k^2 \) \( v^2 \)
\( \hat{\alpha}_{1,3} \)
\( q^3 \) \( k^3 \) \( v^3 \)
\( \hat{\alpha}_{1,4} \)
\( q^4 \) \( k^4 \) \( v^4 \)

\[ \alpha_{1,1} = k^1 q^1 \]
\[ \alpha_{1,2} = k^2 q^1 \]
\[ \alpha_{1,3} = k^3 q^1 \]
\[ \alpha_{1,4} = k^4 q^1 \]

(ignore \( \sqrt{d} \) for simplicity)
Self-attention

\[ b^2 = \sum_i \hat{\alpha}_{2,i} v^i \]
Self-attention

\[ b^2 = \sum_i \hat{\alpha}_{2,i} v^i \]

\[ b^1 b^2 b^3 b^4 = v^1 v^2 v^3 v^4 \]

\[ O = V \hat{A} \]
反正就是一堆矩陣乘法，用 GPU 可以加速
Multi-head Self-attention

\[ q^{i,1} = W^{q,1}q^i \]
\[ q^{i,2} = W^{q,2}q^i \]

(2 heads as example)
Multi-head Self-attention (2 heads as example)

\[ q^{i,1} = W^{q,1} q^i \]
\[ q^{i,2} = W^{q,2} q^i \]

\[ q^i = W^{q} x^i \]
Positional Encoding

- No position information in self-attention.
- Original paper: each position has a unique positional vector $e_i$ (not learned from data)
- In other words: each $x_i$ appends a one-hot vector $p_i$

\[
p^i = \begin{bmatrix} 0 & 1 & 0 & \cdots \end{bmatrix}
\]

\[
p^i = W^L p^i + W^P + e^i + a^i
\]
\[ W^I \mathbf{x}^i + W^P \mathbf{p}^i = x^i \]
Seq2seq with Attention

Encoder

Self-Attention Layer

Decoder

Self-Attention Layer

Review: https://www.youtube.com/watch?v=ZjfjPzXw6og&feature=youtu.be
Using Chinese to English translation as example
**Transformer**

\[ \begin{align*}
    b' & \rightarrow \text{Layer Norm} \\
    b' & + b \\
    b & \rightarrow \\
    a \\
\end{align*} \]

**Layer Norm:**
https://arxiv.org/abs/1607.06450

**Batch Norm:**
https://www.youtube.com/watch?v=BZh1ltr5Rkg

**Batch Size**
\[ \begin{align*}
    \mu &= 0, \quad \sigma = 1 \\
    \text{Batch} \\
\end{align*} \]

**Layer Norm:**
\[ \begin{align*}
    \mu &= 0, \quad \sigma = 1 \\
    \text{Layer} \\
\end{align*} \]

Attend on the input sequence

**Masked:** attend on the generated sequence

**Positional Encoding**

**Input Embedding**

**Output Embedding**

**Add & Norm**

**Multi-Head Attention**

**Softmax**

**Linear**

**Feed Forward**

**Output Probabilities**
Attention Visualization

https://arxiv.org/abs/1706.03762
Attention Visualization

The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

Multi-head Attention
Example Application

- If you can use seq2seq, you can use transformer.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Input</th>
<th>Output</th>
<th># examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gigaword (Graff &amp; Cieri, 2003)</td>
<td>$10^1$</td>
<td>$10^1$</td>
<td>$10^6$</td>
</tr>
<tr>
<td>CNN/DailyMail (Nallapati et al., 2016)</td>
<td>$10^2$–$10^3$</td>
<td>$10^1$</td>
<td>$10^5$</td>
</tr>
<tr>
<td>WikiSum (ours)</td>
<td>$10^2$–$10^6$</td>
<td>$10^1$–$10^3$</td>
<td>$10^6$</td>
</tr>
</tbody>
</table>

https://arxiv.org/abs/1801.10198
Universal Transformer

Parameters are tied across positions and time steps

Self-Attention GAN

convolution feature maps (x)

f(x)

g(x)

h(x)

transpose

softmax

attention map

self-attention feature maps (o)

https://arxiv.org/abs/1805.08318