各式各樣的 Attention
Hung-yi Lee 李宏毅
Prerequisite

https://youtu.be/hYdO9CscNes
【機器學習2021】自注意力機制 (Self-attention) (上)

https://youtu.be/gmsMY5kc-zw
【機器學習2021】自注意力機制 (Self-attention) (下)
To Learn More ...

Long Range Arena: A Benchmark for Efficient Transformers

Efficient Transformers: A Survey
How to make self-attention efficient?

Sequence length = \( N \)

**Attention Matrix**

\[ \text{query} \]

\[ \text{key} \]

\[ N \times N \]
Notice

• Self-attention is only a module in a larger network.

• Self-attention dominates computation when $N$ is large.

• Usually developed for image processing

\[
N = 256 \times 256
\]
Skip Some Calculations with Human Knowledge

Can we fill in some values with human knowledge?
Local Attention / Truncated Attention

Calculate attention weight
Set to 0

Similar with CNN

Calculate attention weight
Stride Attention
Global Attention

special token = “token 中的里長伯“

Add special token into original sequence
- Attend to every token \( \rightarrow \) collect global information
- Attended by every token \( \rightarrow \) it knows global information

No attention between non-special token
Many Different Choices ...

Different heads use different patterns.

小孩子才做選擇・・・

Different heads use different patterns.
Many Different Choices ...

• **Longformer**
  
  ![Sliding window attention](https://arxiv.org/abs/2004.05150)
  ![Dilated sliding window](https://arxiv.org/abs/2007.14062)

• **Big Bird**
  
  ![Random attention](https://arxiv.org/abs/2007.14062)
  ![Window attention](https://arxiv.org/abs/2007.14062)
Can we only focus on Critical Parts?

How to quickly estimate the portion with small attention weights?

- Directly set to 0
- Smaller influence on results
Clustering

Reformer
https://openreview.net/forum?id=rkgNKkHtvB

Routing Transformer

**Step 1**

Clustering based on similarity (approximate & fast)
### Clustering

#### Step 2

<table>
<thead>
<tr>
<th>query</th>
<th>key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Belong to the same cluster, then calculate attention weight**
- **Not the same cluster, set to 0**
Learnable Patterns
Sinkhorn Sorting Network

A grid should be skipped or not is decided by another learned module

Do we need full attention matrix?

Linformer

Many redundant columns

Low Rank
Can we reduce the number of queries?

Can we change output sequence length?
Reduce Number of Keys

Compressed Attention
https://arxiv.org/abs/1801.10198

Linformer

Linear combination of $N$ vectors
Attention Mechanism is three-matrix Multiplication

Review

\[ W_q \cdot Q = W^q \cdot I \]
\[ W_k \cdot K = W^k \cdot I \]
\[ W_v \cdot V = W^v \cdot I \]

\[ A' \xrightarrow{\text{softmax}} A \xrightarrow{\text{ignore}} K^T \cdot Q \]

\[ O = V \cdot A' \]
Attention Mechanism is three-matrix Multiplication

Review

\[ \begin{align*}
W^q W^k W^v & \quad = \quad I \\
Q K V & \quad = \quad I \\
\end{align*} \]

\[ O \approx V K^T Q \]

\[ N \times d \]
What is the difference?
Review Linear Algebra

Practical Issue

- Let $A$ and $B$ be $k \times m$ matrices, $C$ be an $m \times n$ matrix, and $P$ and $Q$ be $n \times p$ matrices.
  - $A(CP) = (AC)P$

- $k = 1$, $m = 1000$, $n = 1$, $p = 1000$

https://youtu.be/yO8lDzf4jMs
\[
O \approx V K^T Q
\]

\[
(d + d') N^2
\]
\[ Q \approx V K^T Q \]

\[ d' \times N \]

\[ d' \times N \]

\[ d \times N \]

\[ N \times d \]

\[ d' \times N \times d \]

\[ d' \times d \]

\[ ? \]

\[ Q \]

\[ d \times N \]

\[ d' \times d \times N \]

\[ 2d' \times dN \]
\( K \) \( T \) \( Q \) 

\[ O \approx V \approx (d + d')N^2 \]

\[ d \times N \]

\[ d \times N \]

\[ d \times N \]

\[ N \times d \]

\[ N \times d \]
Let’s put softmax back ...

Warning of math
$$b^1 = \sum_{i=1}^{N} \alpha'_{1,i} v^i = \sum_{i=1}^{N} \frac{\exp(q^1 \cdot k^i)}{\sum_{j=1}^{N} \exp(q^1 \cdot k^j)} v^i$$
\[ b^1 = \sum_{i=1}^{N} \alpha'_{1,i} v^i = \sum_{i=1}^{N} \frac{\exp(q^1 \cdot k^i)}{\sum_{j=1}^{N} \exp(q^1 \cdot k^j)} v^i \]

\[ \exp(q \cdot k) \approx \phi(q) \cdot \phi(k) \]

\[ q \rightarrow \phi \rightarrow \phi(q) \]

\[ \phi(q^1) \cdot \sum_{j=1}^{N} \phi(k^j) \]

\[ \frac{\sum_{i=1}^{N} [\phi(q^1) \cdot \phi(k^i)] v^i}{\sum_{j=1}^{N} \phi(q^1) \cdot \phi(k^j)} \]
\[ b^1 = \sum_{i=1}^{N} \alpha'_{1,i} \nu^i = \frac{\sum_{i=1}^{N} [\phi(q^1) \cdot \phi(k^i)] \nu^i}{\phi(q^1) \cdot \sum_{j=1}^{N} \phi(k^j)} \]

\[ \phi(q^1) = \begin{bmatrix} q_1^1 \\ q_2^1 \\ \vdots \end{bmatrix} \quad \phi(k^1) = \begin{bmatrix} k_1^1 \\ k_2^1 \\ \vdots \end{bmatrix} \]

\[ = [\phi(q^1) \cdot \phi(k^1)] \nu^1 + [\phi(q^1) \cdot \phi(k^2)] \nu^2 + \cdots \]

\[ = (q_1^1 k_1^1 + q_2^1 k_2^1 + \cdots) \nu^1 + (q_1^1 k_1^2 + q_2^1 k_2^2 + \cdots) \nu^2 + \cdots \]

\[ = q_1^1 k_1^1 \nu^1 + q_1^1 k_2^1 \nu^1 + \cdots + q_1^1 k_1^2 \nu^2 + q_1^2 k_2^2 \nu^2 + \cdots + \cdots \]

\[ = q_1^1 (k_1^1 \nu^1 + k_1^2 \nu^2 + \cdots) + q_2^1 (k_2^1 \nu^1 + k_2^2 \nu^2 + \cdots) \]
\[ b^1 = \sum_{i=1}^{N} \alpha'_{1,i} v^i = \frac{\sum_{i=1}^{N} \left[ \phi(q^1) \cdot \phi(k^i) \right] v^i}{\phi(q^1) \cdot \sum_{j=1}^{N} \phi(k^j)} \]

\[ \sum_{i=1}^{N} \left[ \phi(q^1) \cdot \phi(k^i) \right] v^i \]

\[ \phi(q^1) = \begin{bmatrix} q_1^1 \\ q_2^1 \\ \vdots \end{bmatrix} \quad \phi(k^1) = \begin{bmatrix} k_1^1 \\ k_2^1 \\ \vdots \end{bmatrix} \]

\[ = q_1^1 (k_1^1 v^1 + k_2^1 v^2 + \cdots) + q_2^1 (k_2^1 v^1 + k_2^2 v^2 + \cdots) \]

\[ \sum_{j=1}^{N} k_j^1 v_j \]

\[ \sum_{j=1}^{N} k_j^2 v_j \]

M vectors
\[
\sum_{j=1}^{N} k_1^j \nu^j \quad \sum_{j=1}^{N} k_2^j \nu^j
\]

\[
b_1 = \sum_{j=1}^{N} \phi(k^j)
\]

M vectors

M dim
Don’t compute again
Let’s put softmax back ...

End of warning
\[ \phi(q^1) = \begin{bmatrix} q_1^1 \\ q_2^1 \\ \vdots \end{bmatrix} \]

\[ \phi(k^1) = \begin{bmatrix} k_1^1 \\ v_1^1 \\ q_2^1 \\ k_2^1 \\ \vdots \end{bmatrix} \]

\[ \phi(k^2) = \begin{bmatrix} a_1^1 \\ k_2^2 \\ v_2^2 \\ q_2^2 \\ k_3^2 \\ \vdots \end{bmatrix} \]

\[ \phi(k^3) = \begin{bmatrix} a_3^3 \\ k_3^3 \\ v_3^3 \\ q_2^3 \\ k_4^3 \\ \vdots \end{bmatrix} \]

\[ \phi(k^4) = \begin{bmatrix} a_4^4 \\ k_4^4 \\ v_4^4 \\ q_4^4 \\ \vdots \end{bmatrix} \]

weighted sum

\[ \begin{bmatrix} \phi(q^1) \\ \phi(k^1) \\ \phi(k^2) \\ \phi(k^3) \\ \phi(k^4) \end{bmatrix} = \begin{bmatrix} v_1^1 + v_2^2 + v_3^3 + v_4^4 \end{bmatrix} \]

\[ \vdots \]

M vectors

M dimensions

\[ \phi(q^1) = \begin{bmatrix} q_1^1 \\ q_2^1 \\ \vdots \end{bmatrix} \]
\[ \phi(q^1) = \begin{bmatrix} q_1^1 \\ q_2^1 \\ \vdots \end{bmatrix} \text{ weighted sum } = \begin{bmatrix} \nu^1 \\ \nu^2 \\ \nu^3 \\ \nu^4 \end{bmatrix} \]

\[ b^1 = \frac{\sum_{j=1}^{N} \phi(k^j)}{\phi(q^1)} \]
\[ b_2 = ? \]

Template selection

\( \mathbf{a}_1 \)

\( q_1 \)

\( k_1 \)

\( v_1 \)

\( \phi(k_1) \)

\( \mathbf{a}_2 \)

\( q_2 \)

\( k_2 \)

\( v_2 \)

\( \phi(k_2) \)

\( \mathbf{a}_3 \)

\( q_3 \)

\( k_3 \)

\( v_3 \)

\( \phi(k_3) \)

\( \mathbf{a}_4 \)

\( q_4 \)

\( k_4 \)

\( v_4 \)

\( \phi(k_4) \)

\[ \mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3 + \mathbf{v}_4 = \phi(q_2) \]

\[ \mathbf{v}_1, \ldots, \mathbf{v}_M \] vectors

Template 1

Template 2
Realization

- Efficient attention

- Linear Transformer
  https://linear-transformers.com/

- Random Feature Attention

- Performer
Do we need q and k to compute attention? Synthesizer!

From q and k?

They are network parameters!

\[
\begin{array}{cccc}
\alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} \\
\alpha_{1,2} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\
\alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{3,4} \\
\alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \\
\end{array}
\]

**softmax**

\[ b^1 = \sum_{i=1}^{N} \alpha'_{1,i} v^i \]
Attention-free?

• Fnet: Mixing tokens with fourier transforms
  https://arxiv.org/abs/2105.03824

• Pay Attention to MLPs
  https://arxiv.org/abs/2105.08050

• MLP-Mixer: An all-MLP Architecture for Vision
  https://arxiv.org/abs/2105.01601
Summary

- Human knowledge
  - Local Attention, Big Bird
- Clustering
  - Reformer
- Learnable Pattern
  - Sinkhorn
- Representative key
  - Linformer
- v, k first → v, k first
  - Linear Transformer, Performer
- New framework
  - Synthesizer