Quantum Self-Attention

Dominic Pasquali

March, 23 2022

2022 CERN openlab Technical Workshop
Agenda

- Motivation for Quantum GANs & Quantum Self-Attention
- Classical Self-Attention
- Quantum Self-Attention
- Multi-Head Attention
Motivation For Quantum GANs & Quantum Self-Attention

Why GANs:
- Simulation of particle transport through matter is fundamental for understanding the physics of High Energy Physics (HEP) experiments
- Most of LHC CPU budget (~ 1M CPU-years!!!) is dedicated to Monte Carlo simulation
- Faster approach: Replace Monte Carlo simulation with deep learning algorithms (e.g. GANs)

Why QGANs:
- compressed data representation in quantum states
- expect faster training with less number of parameters
- potential advantage of Quantum GAN\[1\]

Explore different prototypes of Quantum GANs to improve model
- **Quantum Self-Attention** in Classical GANs to boost performance in hybrid architecture
Classical Self-Attention

Query, Key, Value concept analogous to retrieval systems

Example: When you searching for videos on YouTube’s search engine

- search engine maps the **Query (text in the search bar)** against **Keys**
- **Keys: descriptors** (video title, description, etc.) of YouTube videos
- search engine returns the **best matched videos (Values)**

\[
\begin{align*}
W^Q, W^K, W^V & \text{ learned matrices} \\
q_1 & = \mathbf{W}^Q \mathbf{x} \\
k_1 & = \mathbf{W}^K \mathbf{x} \\
v_1 & = \mathbf{W}^V \mathbf{x}
\end{align*}
\]
Classical Self-Attention

<table>
<thead>
<tr>
<th>Input</th>
<th>Learning</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding</td>
<td>X&lt;sub&gt;1&lt;/sub&gt;</td>
<td>X&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td>Queries</td>
<td>q&lt;sub&gt;1&lt;/sub&gt;</td>
<td>q&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td>Keys</td>
<td>k&lt;sub&gt;1&lt;/sub&gt;</td>
<td>k&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td>Values</td>
<td>v&lt;sub&gt;1&lt;/sub&gt;</td>
<td>v&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
</tbody>
</table>

Adding more words adds more resulting vectors (while using the same learned matrices)

For now, let’s only consider 2 words as our input: “Learning Machines”
Classical Self-Attention

<table>
<thead>
<tr>
<th>Input</th>
<th>Learning</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding</td>
<td>X_1</td>
<td>X_2</td>
</tr>
<tr>
<td>Queries</td>
<td>q_1</td>
<td>q_2</td>
</tr>
<tr>
<td>Keys</td>
<td>k_1</td>
<td>k_2</td>
</tr>
<tr>
<td>Values</td>
<td>v_1</td>
<td>v_2</td>
</tr>
</tbody>
</table>
Classical Self-Attention

Computing output for the word *Learning*

- **Input**
  - Embedding
  - Queries
  - Keys
  - Values
  - Score

- **Learning**
  - Input: $X_1$
  - Queries: $q_1$
  - Keys: $k_1$
  - Values: $v_1$
  - Score: $q_1 \cdot k_1 = 112$

- **Machines**
  - Input: $X_2$
  - Queries: $q_2$
  - Keys: $k_2$
  - Values: $v_2$
  - Score: $q_1 \cdot k_2 = 96$
## Classical Self-Attention

### Computing output for the word *Learning*

<table>
<thead>
<tr>
<th>Input</th>
<th>Learning</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding</td>
<td><img src="#" alt="X" /></td>
<td><img src="#" alt="X" /></td>
</tr>
<tr>
<td>Queries</td>
<td><img src="#" alt="q&lt;sub&gt;1&lt;/sub&gt;" /></td>
<td><img src="#" alt="q&lt;sub&gt;2&lt;/sub&gt;" /></td>
</tr>
<tr>
<td>Keys</td>
<td><img src="#" alt="k&lt;sub&gt;1&lt;/sub&gt;" /></td>
<td><img src="#" alt="k&lt;sub&gt;2&lt;/sub&gt;" /></td>
</tr>
<tr>
<td>Values</td>
<td><img src="#" alt="v&lt;sub&gt;1&lt;/sub&gt;" /></td>
<td><img src="#" alt="v&lt;sub&gt;2&lt;/sub&gt;" /></td>
</tr>
<tr>
<td>Score</td>
<td><img src="#" alt="q&lt;sub&gt;1&lt;/sub&gt; • k&lt;sub&gt;1&lt;/sub&gt; = 112" /></td>
<td><img src="#" alt="q&lt;sub&gt;1&lt;/sub&gt; • k&lt;sub&gt;2&lt;/sub&gt; = 96" /></td>
</tr>
<tr>
<td>Score $\div \sqrt{d_{\text{key}}}$</td>
<td>$112 / \sqrt{d_{\text{key}}} = 14$</td>
<td>$96 / \sqrt{d_{\text{key}}} = 12$</td>
</tr>
</tbody>
</table>

$d_{\text{key}} = \text{dimension of key vector}$
- Leads to more stable gradients
- Hyperparameter (!!!)
- Other values may be used

$d_{\text{key}} = \sqrt{64} = 8$

(8 is the value used in *Attention Is All You Need* (2017))
# Classical Self-Attention

## Input
- **Embedding**
- **Queries**
- **Keys**
- **Values**

## Learning
- Query $q_1$
- Key $k_1$
- Value $v_1$

## Score
$$\text{Score} = \frac{q_1 \cdot k_1}{\sqrt{d_{\text{key}}}}$$
$$112 / \sqrt{d_{\text{key}}} = 14$$

## Softmax
$$s_1 = 0.88$$

## Machines
- Query $q_2$
- Key $k_2$
- Value $v_2$

## Score
$$\text{Score} = \frac{q_1 \cdot k_2}{\sqrt{d_{\text{key}}}}$$
$$96 / \sqrt{d_{\text{key}}} = 12$$

## Softmax
$$s_2 = 0.12$$

$d_{\text{key}}$ = dimension of key vector
- Leads to more stable gradients
- Hyperparameter (!!!)
- Other values may be used

$$\sqrt{d_{\text{key}}} = \sqrt{64} = 8$$

(8 is the value used in the original self-attention paper)
Classical Self-Attention

Computing output for the word *Learning*

<table>
<thead>
<tr>
<th>Input</th>
<th>Learning</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Embedding</strong></td>
<td><img src="image" alt="Embedding" /></td>
<td><img src="image" alt="Embedding" /></td>
</tr>
<tr>
<td><strong>Queries</strong></td>
<td><img src="image" alt="Queries" /></td>
<td><img src="image" alt="Queries" /></td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td><img src="image" alt="Keys" /></td>
<td><img src="image" alt="Keys" /></td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td><img src="image" alt="Values" /></td>
<td><img src="image" alt="Values" /></td>
</tr>
<tr>
<td><strong>Score</strong></td>
<td><img src="image" alt="Score" /></td>
<td><img src="image" alt="Score" /></td>
</tr>
<tr>
<td><strong>Softmax</strong></td>
<td><img src="image" alt="Softmax" /></td>
<td><img src="image" alt="Softmax" /></td>
</tr>
<tr>
<td><strong>Softmax • Value</strong></td>
<td><img src="image" alt="Softmax • Value" /></td>
<td><img src="image" alt="Softmax • Value" /></td>
</tr>
</tbody>
</table>

\( \text{d}_{\text{key}} = \text{dimension of key vector} \)
- Leads to more stable gradients
- Hyperparameter (!!!)
- Other values may be used

\( \sqrt{\text{d}_{\text{key}}} = \sqrt{64} = 8 \)

(8 is the value used in the original self-attention paper)
### Classical Self-Attention

#### Input

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Queries</th>
<th>Keys</th>
<th>Values</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>$q_1$</td>
<td>$k_1$</td>
<td>$v_1$</td>
<td>$s_1$ = 0.88</td>
</tr>
</tbody>
</table>

#### Learning

- **Score**: $q_1 \cdot k_1 = 112$
- **Score $\div \sqrt{d_{\text{key}}}$**: $112 / \sqrt{d_{\text{key}}} = 14$
- **Softmax**: $s_1 = 0.88$

#### Machines

| $X_2$     | $q_2$   | $k_2$ | $v_2$  | $s_2$ = 0.12 |

- **Score**: $q_1 \cdot k_2 = 96$
- **Score $\div \sqrt{d_{\text{key}}}$**: $96 / \sqrt{d_{\text{key}}} = 12$

#### Softmax • Value

- **Softmax • Value $m_1 = s_1 \cdot v_1$**
- **Softmax • Value $m_2 = s_2 \cdot v_2$**

#### Sum

- **Sum $z_1 = m_1 + m_2$**

---

$d_{\text{key}}$ = dimension of key vector

- Leads to more stable gradients
- Hyperparameter (!!!)
- Other values may be used

$\sqrt{d_{\text{key}}} = \sqrt{64} = 8$

(8 is the value used in the original self-attention paper)
Classical Self-Attention

Input
Embedding
Queries
Keys
Values
Score
Score \( \div \sqrt{d_{\text{key}}} \)
Softmax
Softmax \( \times \) Value
Sum

Learning

Machines

This is only for the word *Learning*!

\[
q_1 \cdot k_1 = 112
\]

\[
q_1 \cdot k_2 = 96
\]

\[
s_1 = 0.88
\]

\[
s_2 = 0.12
\]

\[
m_1 = s_1 \cdot v_1
\]

\[
m_2 = s_2 \cdot v_2
\]

\[
z_1 = m_1 + m_2
\]

\[
\sqrt{d_{\text{key}}} = \sqrt{64} = 8
\]

\[d_{\text{key}} = \text{dimension of key vector}
\]

\(\circ\) Leads to more stable gradients

\(\circ\) Hyperparameter (!!!)

\(\circ\) Other values may be used

(8 is the value used in the original self-attention paper)
Classical Self-Attention

### Input
- **Embedding**
- **Queries**
- **Keys**
- **Values**

### Learning
- \( X_1 \)
- \( q_1 \)
- \( k_1 \)
- \( v_1 \)

### Machines
- \( X_2 \)
- \( q_2 \)
- \( k_2 \)
- \( v_2 \)

#### Score
- \( \text{Score} = q_2 \cdot k_1 = 16 \)
- \( 16 / \sqrt{d_{\text{key}}} = 2 \)
- \( s_1 = 0.27 \)

#### Score
- \( \text{Score} = q_2 \cdot k_2 = 24 \)
- \( 24 / \sqrt{d_{\text{key}}} = 3 \)
- \( s_2 = 0.73 \)

#### Softmax
- \( m_1 = s_1 \cdot v_1 \)
- \( m_2 = s_2 \cdot v_2 \)

#### Sum
- \( z_2 = m_1 + m_2 \)

---

Doing this for the word *Machines* is just as easy.

Creating a score with the query from *Machines*.

Summing over each *m* from the new values.
Classical Self-Attention: Another Look
Quantum Self-Attention

Input

Embedding

X

Learning

Queries

q₁

Keys

k₁

Values

v₁

X

1

W_Q

W_K

W_V

W_Q, W_K, W_V

learned matrices quantum circuits!

= q₁

= k₁

= v₁
Multi-Head Attention

Attention Head #0

Queries $Q_0$ $W_0^Q$

Keys $K_0$ $W_0^K$

Values $V_0$ $W_0^V$

Attention Head #1

Queries $Q_1$ $W_1^Q$

Keys $K_1$ $W_1^K$

Values $V_1$ $W_1^V$
Multi-Head Attention

Concatenate $Z$ from each attention head:

$Z_0 + Z_1 + \ldots + Z_7$

Multiply concatenated $Z$'s with learned matrix $W^0$

Resulting matrix $Z$ contains information from all attention heads
Efficient Multi-Head Attention

Stacking words results in a larger matrix

Allows for representing each input as a (larger) matrix

Learning Machines
In < 50 epochs, we get (preliminary) results with MNIST dataset!

Dr. Sofia Vallecorsa

Dr. Michele Grossi

Computational Resources:
CERN openlab
Questions?