Attention and Transformer Models for Genomics

BMI/CS 776
www.biostat.wisc.edu/bmi776/
Spring 2024
Daifeng Wang
daifeng.wang@wisc.edu

Thanks to Noah Cohen Kalafut for slides!

These slides, excluding third-party material, are licensed under CC BY-NC 4.0 by Anthony Gitter, Mark Craven, Colin Dewey and Daifeng Wang
Goals for lecture

• Attention
  – Interpretation
  – Self and Cross-Attention Calculation
  – Multi-Head Attention

• Transformer architecture
  – Positional Encoding
  – Generative Output

• Applications to bioinformatics
Attention Interpretation

scores

Attention Calculation

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

animal

street

it
Attention Calculation

- Embeddings are passed through feed-forward networks to produce a query vector, as well as key and value vectors.

- Value vectors are summed proportionally to the similarity between their corresponding keys and the query.
Attention Calculation

- **Key** and **query** vectors can be thought of as sharing a latent space.
- The distance between the **query** and **keys** then determines the final **output** in **value** space.
Types of Attention

Self-Attention

Cross-Attention
Multi-Head Attention

Each attention module is known as a ‘head’

Multi-head attention involves aggregating multiple heads, usually culminating in average pooling

This allows each head to prioritize differing features of the sequence (e.g. noun-noun relationships, noun-verb relationships)
Transformer Outline

Input Sequence: A C C T A G A A C G

Output Sequence: A C G .. T C G A A T
Example: Generative Text

Find the bug with this code:

```javascript
for (var i = 0; i < 5; i++) {
  setTimeout(function() => console.log(i), 1000)
}
```

The code contains a bug that will cause it to always print the number 5 to the console, rather than the expected sequence of numbers from 0 to 4. This is because the `setTimeout` function creates a closure that captures the value of the `i` variable, but the `i` variable is changed before the callback function is executed.

To fix this bug, the code can be modified to create a new variable in the loop that is passed to the `setTimeout` callback function, rather than using the loop variable directly. This will ensure that each callback function captures the correct value of the loop variable, and the expected output will be produced. Here is an example of how the code could be fixed:

```javascript
for (var i = 0; i < 5; i++) {
  // Create a new variable and pass its value to setTimeout
  const value = i;
  setTimeout(() => console.log(value), 1000)
}
```

This code will print the numbers 0 to 4 to the console, as expected. Alternatively, the `let` keyword can be used in place of `var` to declare the loop variable, which will automatically create a new variable for each iteration of the loop and avoid the need to create a new variable manually. This is a common pattern for avoiding closure-related bugs in JavaScript.

Transformer Architecture

https://www.tensorflow.org/text/tutorials/transformer
Input Embedding

Input Sequence

ACCTAGAACC

Learned Embeddings

0 4 4 5 0 8 0 0 0 8
2 8 8 8 2 3 2 2 8 3
5 5 5 2 5 0 5 5 5 0
3 8 8 8 3 6 3 3 8 6

https://www.tensorflow.org/text/tutorials/transformer
Positional Encoding

Add positional representations to word embeddings

• Allows the network to consider word proximity

https://www.tensorflow.org/text/tutorials/transformer
Positional Encoding

Positional Encoding Matrix for the sequence ‘I am a robot’
**Positional Encoding**

\[ P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right) \]
\[ P(k, 2i + 1) = \cos\left(\frac{k}{n^{2i/d}}\right) \]

---

**Positional Encoding Matrix for the sequence ‘I am a robot’**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Index of token, ( k )</th>
<th>( i=0 )</th>
<th>( i=0 )</th>
<th>( i=1 )</th>
<th>( i=1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0</td>
<td>( P_{00}=\sin(0) = 0 )</td>
<td>( P_{01}=\cos(0) = 1 )</td>
<td>( P_{02}=\sin(0) = 0 )</td>
<td>( P_{03}=\cos(0) = 1 )</td>
</tr>
<tr>
<td>am</td>
<td>1</td>
<td>( P_{10}=\sin(1/1) = 0.84 )</td>
<td>( P_{11}=\cos(1/1) = 0.54 )</td>
<td>( P_{12}=\sin(1/10) = 0.10 )</td>
<td>( P_{13}=\cos(1/10) = 1.0 )</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>( P_{20}=\sin(2/1) = 0.91 )</td>
<td>( P_{21}=\cos(2/1) = -0.42 )</td>
<td>( P_{22}=\sin(2/10) = 0.20 )</td>
<td>( P_{23}=\cos(2/10) = 0.98 )</td>
</tr>
<tr>
<td>Robot</td>
<td>3</td>
<td>( P_{30}=\sin(3/1) = 0.14 )</td>
<td>( P_{31}=\cos(3/1) = -0.99 )</td>
<td>( P_{32}=\sin(3/10) = 0.30 )</td>
<td>( P_{33}=\cos(3/10) = 0.96 )</td>
</tr>
</tbody>
</table>
Positional Encoding

Learned Embeddings
0 4 4 5 0 8 0 0 0 8
2 8 8 8 2 3 2 2 8 3
5 5 5 2 5 0 5 5 5 0
3 8 8 8 3 6 3 3 8 6

Positional Encoding

Temporal Embeddings
0 6 0 6 0 2 3 8 5 9
2 2 2 0 9 7 4 9 1 2 7
4 7 1 5 3 3 7 6 3 1
3 5 7 1 2 5 3 0 3 9

https://www.tensorflow.org/text/tutorials/transformer
Attention Modules

Prioritize bases relative to each other

- This is the primary mechanism which allows transformers to work
- Essentially adds context to existing embeddings
Self and Cross-Attention

Self-attention (Q=K=V)

Cross-attention (Q from decoder) (K=V from encoder)

Self-attention (Q=K=V)

https://www.tensorflow.org/text/tutorials/transformer
Transformer Attention

Temporal Embeddings

0 6 0 6 0 2 3 8 5 9
2 2 2 0 9 7 4 9 1 2 7
4 7 1 5 3 3 7 6 3 1
3 5 7 1 2 5 3 0 3 9

Encoded Representations

0 5 7 5 5 2 3 7 7 0
5 6 1 8 1 2 3 1 2 4
0 8 3 7 6 1 7 8 8 3
2 7 7 3 9 0 0 4 0 0

Multi-Head Attention + Feed-Forward

https://www.tensorflow.org/text/tutorials/transformer
Transformer Decoder

For generative outputs, repeatedly choose the most likely next element until the end of the sequence

0. <start>
1. <start> A
2. <start> A T
3. <start> A T T
4. <start> A T T G
5. <start> A T T G <end>

https://www.tensorflow.org/text/tutorials/transformer
Example: Protein Function Annotation
Example: Enhancer Prediction from DNA Sequence

Transformers have applications other than sequence generation as well

A C C T A G A A C G

Enformer
Example: Enhancer Prediction from DNA Sequence

Instead of generating a DNA sequence, genomic tracks (e.g. TF binding, accessibility) can be generated instead

- Convolve 100kb to produce features for each base with attention pooling
- Feed to multiple self-attention blocks (transformer encoder)
- Apply final convolutions to predict tracks for humans or mice
Example: Enhancer Prediction from DNA Sequence

Transformers allow for broader search regions with fewer computational limits.

Predict attribute-correlated locations based on DNA sequence.
Example: Enhancer Prediction from DNA Sequence

After predicting tracks, known enhancers line up with calculated attention scores.