Numpy - Attention score

Hoàng-Nguyên Vũ

1 Giới thiệu về Transformer?

Trong lĩnh vực trí tuệ nhân tạo, **Transformer** là một kiến trúc mạnh mẽ và hiệu quả trong việc xử lý ngôn ngữ tự nhiên (NLP). Một trong những yếu tố quan trọng tạo nên sức mạnh của Transformer là cơ chế Attention. Bài viết này sẽ giúp các bạn mới học về NumPy hiểu rõ cơ chế Attention trong Transformer thông qua các phép toán cơ bản.

2 Attention là gì?

Attention cho phép mô hình tập trung vào các phần khác nhau của chuỗi đầu vào khi tạo ra mỗi phần tử của chuỗi đầu ra. Điều này giúp mô hình đánh giá tầm quan trọng của mỗi từ trong câu và cải thiện khả năng hiểu ngữ cảnh.

3 Công thức Attention

Công thức cho cơ chế Attention được biểu diễn như sau:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Trong đó:

- Q là ma trận query
- K là ma trận key
- V là ma trận value
- d_k là kích thước chiều của vector key
- softmax là hàm softmax, được tính toán như sau:

$$\text{Hàm kích hoạt Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2}$$

4 Chi tiết từng bước

4.1 Tính toán ma trận điểm số

Ma trận điểm số được tính bằng cách nhân ma trận query Q với ma trận key K chuyển vị, sau đó chia cho căn bậc hai của kích thước chiều của vector key d_k:

$$\text{scores} = \frac{QK^T}{\sqrt{d_k}} \tag{3}$$

4.2 Áp dụng hàm Softmax

Sau khi tính toán ma trận điểm số, chúng ta áp dụng hàm softmax để chuyển đổi các điểm số thành các trọng số xác suất:

$$\text{attention\_weights} = \text{softmax}(\text{scores}) \tag{4}$$

4.3 Tính toán tổng có trọng số

Cuối cùng, chúng ta nhân các trọng số attention với ma trận value V để có được đầu ra của cơ chế Attention:

$$\text{output} = \text{attention\_weights} \cdot V \tag{5}$$

5 Kết luận

Công thức Attention trong Transformer giúp mô hình tập trung vào các phần quan trọng của chuỗi đầu vào khi tạo ra đầu ra, giúp cải thiện khả năng hiểu ngữ cảnh của mô hình. Việc hiểu và áp dụng công thức này là cơ sở để làm việc với các mô hình ngôn ngữ hiện đại.

6 Bài tập

Hãy viết hàm tính attention score của câu đầu vào: "Tôi thích học AI" với tập Vocab được cho sẵn như sau:

```python
import numpy as np

# Giúp kết quả giống nhau trên mọi máy khi tạo random
np.random.seed(42)

# Bước 1: Tạo từ điển và mã hóa từ
vocab = {
    "Tôi": 0,
    "thích": 1,
    "học": 2,
    "AI": 3
}

# Số lượng từ vựng
vocab_size = len(vocab)

# Kích thước vector embedding
embedding_dim = 4
```

```

# Khởi tạo ma trận embedding ngẫu nhiên
embedding_matrix = np.random.rand(vocab_size, embedding_dim)

# Chuỗi đầu vào được mã hóa thành các vector embedding
input_seq = np.array([embedding_matrix[vocab[word]] for word in ["Tôi", "
    thích", "học", "AI"]])
print("Chuỗi đầu vào (đã mã hóa):\n", input_seq)

# Bước 2: Khởi tạo các ma trận trọng số cho Q, K, V
W_q = np.random.rand(embedding_dim, embedding_dim)
W_k = np.random.rand(embedding_dim, embedding_dim)
W_v = np.random.rand(embedding_dim, embedding_dim)

# Tính toán Q, K, V
Q = np.dot(input_seq, W_q)
K = np.dot(input_seq, W_k)
V = np.dot(input_seq, W_v)

print("Ma trận Query Q:\n", Q)
print("Ma trận Key K:\n", K)
print("Ma trận Value V:\n", V)

# Bước 3: Tính toán Attetion score
scores = # Your code here #

# Chia cho căn bậc hai của kích thước chiều của vector key
d_k = # Your code here #
scores = scores / np.sqrt(d_k)

print("Điểm số:\n", scores)

# Bước 4: Áp dụng hàm softmax
def softmax(x):
    # Your code here #

attention_weights = softmax(scores)

print("Trọng số Attention:\n", attention_weights)

# Bước 5: Tính toán tổng có trọng số của các value
output = # Your code here #

print("Đầu ra:\n", output)

```

Kết quả:

```
Đầu ra:
[[1.1796727  0.9217873  1.45815851 1.33770808]
 [1.09435999 0.83851284 1.39301108 1.243537  ]
 [1.1515616  0.89314997 1.43845963 1.30643711]
 [1.11065168 0.85384713 1.40560424 1.26208253]]
```