De la Red Neuronal al GPT

Videos de YouTube

Andrej Karpathy - Construyamos GPT: desde cero, en código, explicado paso a paso.

Umar Jamil - La atención es todo lo que necesitas (Transformer) - Explicación del modelo (incluyendo matemáticas), Inferencia y Entrenamiento

StatQuest con Josh Starmer - Redes Neuronales Transformer, la base de ChatGPT, ¡Explicado claramente!!!

Pascal Poupart - CS480/680 Clase 19: Atención y Redes Transformer

El Pirata de la I.A. - Michael Phi - Guía Ilustrada de Redes Neuronales Transformers: Una explicación paso a paso

Cómo Aprendo

Una vez que había leído la mitad del libro “Neural Networks and Deep Learning”, comencé a replicar el ejemplo de la red neuronal para reconocer dígitos escritos a mano. Creé un repositorio en GitHub, https://github.com/lzwjava/neural-networks-and-zhiwei-learning.

Esa es la parte realmente difícil. Si uno puede escribirlo desde cero sin copiar ningún código, entonces lo entiende muy bien.

Mi código de replicación aún carece de la implementación de `update_mini_batch` y `backprop`. Sin embargo, al observar detenidamente las variables en la fase de carga de datos, propagación hacia adelante y evaluación, obtuve una comprensión mucho mejor del vector, la dimensionalidad, la matriz y la forma de los objetos.

Y comencé a aprender la implementación del GPT y el transformer. A través de la incrustación de palabras y la codificación posicional, el texto se convierte en números. Luego, en esencia, no tiene diferencia con una red neuronal simple para reconocer dígitos escritos a mano.

La conferencia de Andrej Karpathy “Let’s build GPT” es muy buena. Explica las cosas de manera clara.

La primera razón es que realmente es desde cero. Primero vemos cómo generar el texto. Es algo difuso y aleatorio. La segunda razón es que Andrej podía explicar las cosas de manera muy intuitiva. Andrej trabajó en el proyecto nanoGPT durante varios meses.

Acabo de tener una nueva idea para juzgar la calidad de la conferencia. ¿Realmente el autor puede escribir estos códigos? ¿Por qué no los entiendo y qué tema está omitiendo el autor? Además de estos elegantes diagramas y animaciones, ¿cuáles son sus deficiencias y defectos?

Volviendo al tema del aprendizaje automático en sí. Como menciona Andrej, el dropout, la conexión residual, la Self-Attention, la Multi-Head Attention y la Masked Attention.

Al ver más de los videos anteriores, comencé a entender un poco.

Mediante la codificación posicional con funciones seno y coseno, obtenemos algunos pesos. A través de la incrustación de palabras, convertimos las palabras en números.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

La pizza salió del horno y sabía bien.

En esta oración, ¿cómo sabe el algoritmo si se refiere a “pizza” o a “horno”? ¿Cómo calculamos las similitudes para cada palabra en la oración?

Queremos un conjunto de pesos. Si utilizamos la red de transformadores para realizar la tarea de traducción, cada vez que ingresamos una oración, puede generar la oración correspondiente en otro idioma.

Sobre el producto punto aquí. Una razón por la que usamos el producto punto aquí es que el producto punto considerará cada número en el vector. ¿Qué pasa si usamos el producto punto al cuadrado? Primero calculamos el cuadrado de los números y luego los hacemos hacer el producto punto. ¿Qué pasa si hacemos un producto punto inverso?

Sobre el enmascaramiento aquí, cambiamos los números de la mitad de la matriz a menos infinito. Luego usamos softmax para que los valores estén en el rango de 0 a 1. ¿Qué tal si cambiamos los números de la parte inferior izquierda a menos infinito?

Plan

Continuar leyendo código y artículos, y ver videos. Simplemente divertirme y seguir mi curiosidad.

https://github.com/karpathy/nanoGPT

https://github.com/jadore801120/attention-is-all-you-need-pytorch