Introducción a la cuantización

Ana López Palomo y Andrea Manuel Simón

October 2023

1 Investigación previa

La cuantización es un proceso mediante el cual se reduce la precisión de los valores numéricos en un modelo de lenguaje o red neuronal, para disminuir los requisitos de memoria y cómputo. De esta manera, se pueden comprimir modelos pre-entrenados y hacer que ocupen menos espacio en disco.

Normalmente, el tamaño de un proceso se determina multiplicando el número de parámetros por la precisiónd e los valores, pero para lograr esta reducción, muchas veces se resringen los valores de los parámetros del modelo a un número menor de bits. Por ejemplo, en vez de utilizar 32 bits para representar un número real, se pueden utilizar 8 y se usa menos memoria para almacenar ese dato.

Existen muchos enfoques para la cuantización, como el redondeo o el truncamiento, pero todos son una manera de determinar cómo se aproxima un número real a uno cuantizado.

La cuantización posterior al entrenamiento (PTQ) y el entrenamiento consciente de la cuantización (QAT) son dos enfoque diferentes de la cuantización en modelos de aprendizaje automático, como las redes neuronales.

1.1 Diferencias entre PTQ y QAT

PTQ: se aplica la cuantización después de terminar el entrenamiento del modelo, así ahorramos espacio en la memoria. Este método es menos flexible, dado que debemos de modificar un modelo que ya ha sido entrenado bajo unos criterios en específico. En algunos casos es necesario ajustar y optimizar tras su cuantización para que rinda de una manera eficiente. Nos proporciona menos coste computacional al no tener operaciones de cuantización durante el entrenamiento.

QAT: la cuantización se relaiza durante el entrenamiento del modelo, aplicando operaciones de cuantización en las distintas capas del modelo. Todo ello permite que se siga el modelo de cuantización mientras se entrena. Es más flexible que PTQ, dado que no hay necesidad de volver a ajustar la configuración del modelo. También necesita menos ajustes para mantener un buen rendimiento, auqnue la cuantización durante el proceso hace que se tarde más y necesitemos más recursos computacionales.

2 Configuración inicial

En el código implementado hemos creado el modelo GPT preentrenado, y hemos mostrado 3 resultados: el texto generado sin cuantizar, el cuantizado por absmax y el cuantizado por zeropoint.

```
# Set device to CPU for now

device = 'cpu'



# Load model and tokenizer

model_id = 'gpt2'

model = AutoModelForCausalLM.from_pretrained(model_id).to(device)

tokenizer = AutoTokenizer.from_pretrained(model_id)
```

Figure 1: Creación del modelo

3 Representación en coma flotante

En las aplicaciones que necesitan un aprendizaje profundo es necesario equilibrar la precisión y el rendimiento computacional. Los datos que mejor se adaptan a este equilibrio son los números punto coma flotante. Estos números usan nortelosbits para almacenar un valor numérico. Estos nortelosbits se dividen en:

-signo: es un bit que indica el signo del número, donde 0 es positivo y 1 es negativo.

-exponente: es un conjunto de bits que indican la potencia que está elevada la base. Este exponente puede ser positivo o negativo.

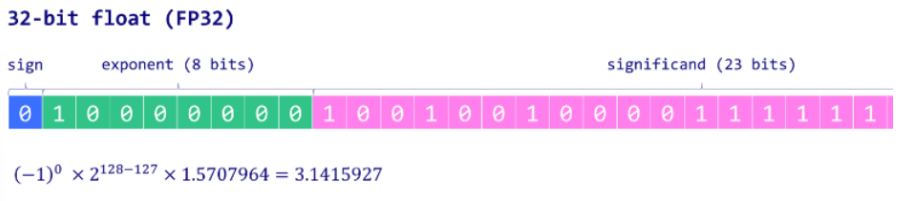
-mantisa/significado: son el resto de bits, estos almacenan el significado. La precisión del número depende bastante de la longitud de estos bits.

Para representar estos números se utiliza: $(-1)^{signo} \times (2^{exponente}) \times mantisa$.

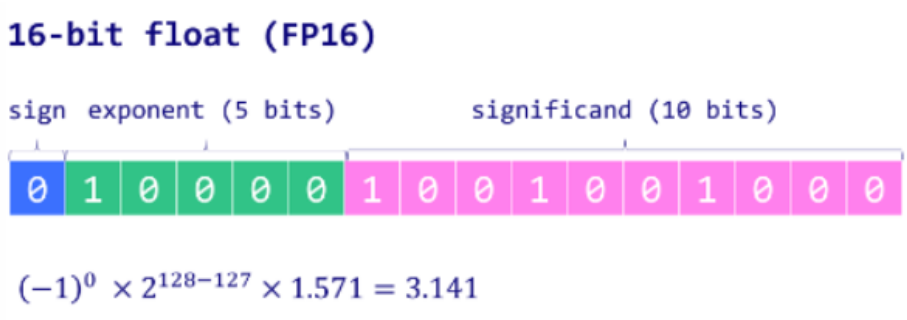
Para comprender esto mejor vamos a ver algunos de los datos que se utilizan en aprendizaje profundo:

-FP32(float32): se compone de 32 bits (1 bit el signo, 8 bits el exponente y 23 bits la mantisa). Son bastantes precisos con un rango amplio y dinámico, aunque consume bastante de memoria y computación debido a sis 32 bits.

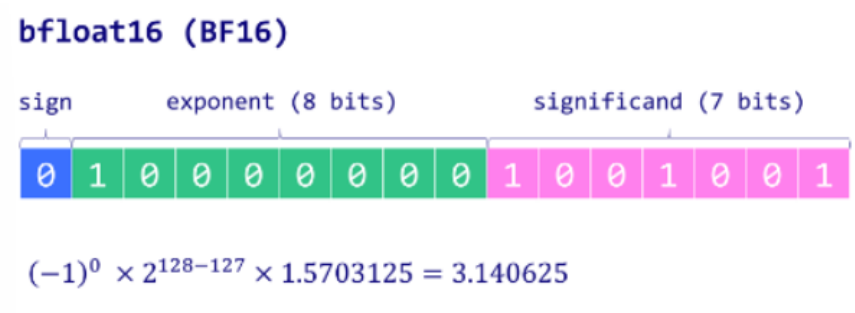
-FP16(float16): se compone de 16 bits(1 bit el signo, 5 bits el exponente y 10 para la mantisa). Consume menos memoria al tener menos bits que el anteiror,



sin embargo, al tener un menor rango y precisión puede generar inestabilidad numérica.



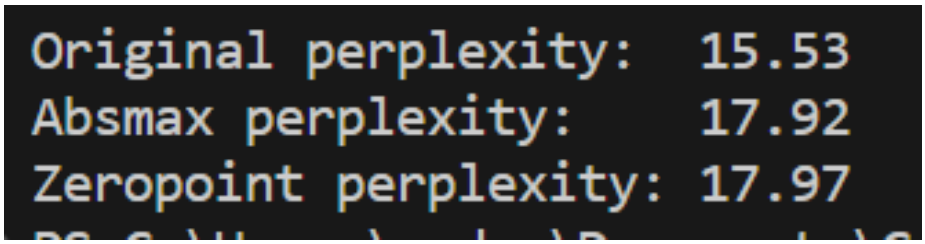
-BF16(float16): se compone de 16 bits(1 bit el signo, 8 bits el exponente y 7 la mantisa) que reduce el riesgo de subdesbordamiento y desbordamiento. Tiene poca precisión debido a la reducción de bits en la mantisa, no obtante, el rendimiento no suele verse afectado por dicha reducción. Su rango es similar al de FP16.



4 Implementación PTQ

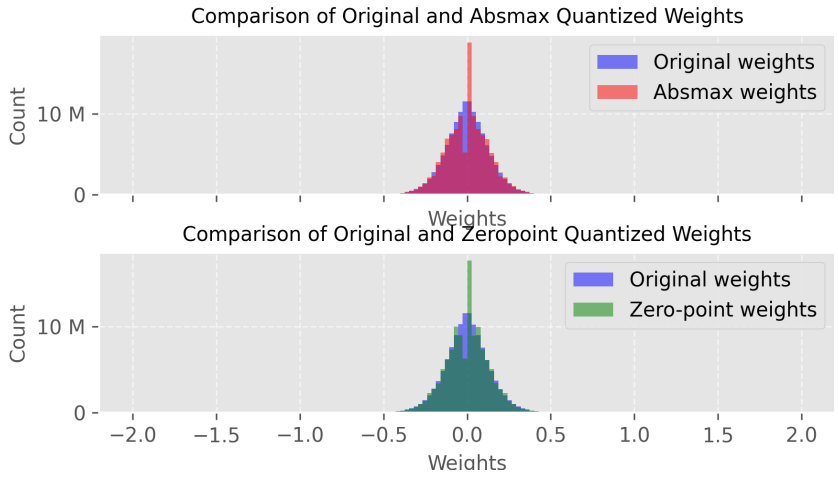
Cuando evaluamos el modelo cuantizado debemos de comparar su perplejidad con el original.

La perplejidad es una medida que se utiliza para evaluar la calidad de los modelos de lenguaje, donde un valor más bajo indica un mejor rendimiento. En general, se espera que la cuantización de los pesos afecte la precisión del modelo, lo que podría reflejarse en una mayor perplejidad en comparación con el modelo original.

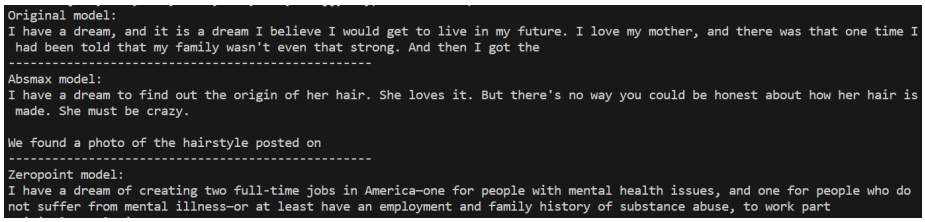


En la imagen se puede observar que la perplejidad del modelo original, sin cuantizar, es menor. Es decir, el modelo original tiene una mayor precisión, pero a cambio ocupa más espacio que sus compañeros cuantizados.

5 Análisis y conclusiones



En la gráfica podemos observar como los pesos, tanto en el modelo absmax como en el modelo zero-point, son menos uniformes, y no siguen la distribución normal que tiene el modelo original. Los pesos en los modelos cuantizados tienden a tomar valores más dispersos, obteniendo un valor atípico.



En conclusión, merece la pena tener una menor precisión a cambio de un menor tamaño y un mejor rendimiento, siempre intentado conseguir el mayor equilibrio posible. Es decir, creemos que el modelo cuantizado da mejores resultados que el original.