Apuntes Semana 14 - 27/05/2025

Victoria Sandí Barrantes c.2022146536

***Abstract*—These notes correspond to week 14 of the course, focusing on two main topics: quantization in deep learning models and unsupervised learning techniques. First, quantization methods (symmetric and asymmetric) are analyzed, including their mathematical formulation, advantages (memory reduction, faster inference), and implementation strategies (Post-Training Quantization and Quantization-Aware Training). Next, unsupervised learning is introduced, with emphasis on clustering algorithms such as K-Means, along with their applications in dimensionality reduction and anomaly detection.**

I. Quantization

A. ¿Qué es Quantization?

En el ámbito del Deep Learning, donde es común encontrar modelos con numerosas capas y parámetros, el almacenamiento y procesamiento de estos parámetros presenta desafíos significativos. Por ejemplo, modelos avanzados como LLaMA-2 llegan a contener hasta 70 mil millones de parámetros. Si consideramos que cada parámetro ocupa 32 bits, el requerimiento de almacenamiento supera los 286GB, lo que dificulta considerablemente su carga en memoria y ralentiza las operaciones de punto flotante.

B. Solución

La técnica de quantization emerge como solución a estos problemas, reduciendo el número de bits necesarios para representar cada parámetro mediante la conversión de valores de punto flotante a enteros. Este enfoque permite comprimir los modelos significativamente, pudiendo representar los parámetros con 8 bits, 5 bits, 2 bits o incluso 1 bit. Es importante destacar que este proceso no consiste simplemente en redondear los pesos a valores enteros, sino que involucra cálculos más complejos para preservar la funcionalidad del modelo.

II. Ventajas de Quantization

A. Beneficios principales

La cuantización ofrece ventajas significativas en el despliegue de modelos de aprendizaje automático, particularmente en entornos con restricciones de recursos:

- **Menor consumo de memoria:**
  - La reducción en precisión numérica (de 32 bits a 8 bits o menos) disminuye drásticamente los requisitos de almacenamiento.
  - Esta compresión permite cargar modelos complejos en dispositivos con memoria limitada como móviles, microcontroladores o sistemas embebidos.
  - Facilita el despliegue simultáneo de múltiples modelos en un mismo dispositivo.
- **Menor tiempo de inferencia:**
  - Las operaciones con enteros son computacionalmente más eficientes que las de punto flotante.
  - Los procesadores modernos incluyen instrucciones especializadas que aceleran las operaciones cuantizadas.
  - La reducción en el ancho de banda de memoria permite alimentar más rápido las unidades de procesamiento.
- **Menor consumo de energía:**
  - Las operaciones con menor precisión requieren menos ciclos de reloj y menos actividad de los circuitos.
  - En dispositivos móviles, esto se traduce en mayor duración de batería.
  - En centros de datos, reduce costos operativos y huella de carbono.
  - La eficiencia energética es crucial para aplicaciones IoT donde los dispositivos funcionan con baterías por años.

III. Representación numérica en sistemas digitales

Los sistemas computacionales representan toda información mediante bits, lo que impone limitaciones y oportunidades para la optimización:

- **Fundamentos de representación binaria:**
  - Cada bit puede tomar valores 0 o 1, siendo N bits capaces de representar 2^N valores distintos
  - Ejemplo básico con 3 bits:

Binario	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

- **Organización práctica en bytes:**
  - La mayoría de sistemas modernos agrupan bits en bloques de 8 (1 byte)
  - Un byte puede representar 256 valores distintos (0-255 sin signo)
  - Esta granularidad afecta directamente el almacenamiento y transferencia de datos
- **Impacto en la cuantización:**
  - La cuantización asigna rangos de valores continuos a estas representaciones discretas
  - El error de cuantización depende de cómo se mapeen los valores reales al espacio discreto

  - Estrategias como el quantization simétrico/asimétrico optimizan este mapeo

La representación numérica óptima depende del contexto de aplicación, donde se debe balancear precisión contra eficiencia computacional.

IV. Redes Neuronales - Pesos

Las matrices de pesos son representadas con punto flotante. Se busca lo siguiente:

- No perder calidad en los resultados cuando se hace quantization
- Procesar todas las operaciones utilizando aritmética de enteros
- La salida de las capas no cambien por aplicar quantization
- Mantener el rendimiento del modelo pero utilizando enteros
- Esto es beneficioso para la mayoría de los hardware especialmente embebidos
- Las capas siguientes no se den cuenta que las operaciones ejecutadas fueron hechas con enteros

V. Quantization Simétrico y Asimétrico

A. Quantization Asimétrico

El quantization asimétrico permite mapear valores en un rango arbitrario $[\beta, \alpha]$ a una representación entera discreta $[0, 2^n - 1]$. Este método es particularmente útil cuando la distribución de los valores no es simétrica alrededor del cero.

- n el número de bits de cuantización
- $\beta =$ el número menor de los valores a convertir
- $\alpha =$ el número mayor de los valores a convertir

Este proceso se realiza mediante una operación de clamp que asegura que los valores cuantizados permanezcan dentro del rango objetivo.

La transformación se realiza mediante:

$$x_q = \text{clamp}\left(\left\lfloor \frac{x_f}{s} \right\rfloor + z; 0; 2^n - 1\right) \tag{1}$$

Donde:

- x_f: Valor original en punto flotante
- x_q: Valor cuantizado entero
- s: Factor de escala calculado como:

$$s = \frac{\alpha - \beta}{2^n - 1} \tag{2}$$

- z: Punto cero que alinea los rangos:

$$z = \left\lfloor -\frac{\beta}{s} \right\rfloor \tag{3}$$

B. Quantization Simétrico

Por otro lado, el quantization simétrico mapea valores en el intervalo [-a, a] a un rango simétrico alrededor de cero $[-(2^n - 1), (2^n - 1)]$.

- $\alpha =$ mayor número absoluto

En este caso, la fórmula de quantization no requiere parámetro de desplazamiento. La dequantization se realiza simplemente multiplicando el valor cuantizado por el factor de escala.

La transformación se realiza mediante:

$$x_q = \text{clamp}\left(\left\lfloor \frac{x_f}{s} \right\rfloor; -(2^{n-1} - 1); 2^{n-1} - 1\right) \tag{4}$$

Donde:

- x_f: Valor original en punto flotante
- x_q: Valor cuantizado entero
- s: Factor de escala calculado como:

$$s = \frac{abs(a)}{2^{n-1} - 1} \tag{5}$$

VI. Dynamic Quantization y Calibration

A. Dynamic Quantization

1) Proceso de quantization: El dynamic quantization busca ejecutar todas las operaciones con enteros:

- **Pesos (W)** y **bias (B)** están pre-cuantizados
- **Entrada (X)** se cuantiza en tiempo de ejecución
- Operación original:

$$Y_f = XW + B$$

2) Retos en la inferencia:

- La salida cuantizada Y_q contiene valores enteros
- La siguiente capa requiere valores en punto flotante
- Problema: No se conocen previamente:
  - El parámetro escalador (s)
  - El zero point (z)

3) Solución propuesta:

- Las activaciones se cuantizan dinámicamente calculando:
  - α (valor máximo observado)
  - β (valor mínimo observado)
- Estos parámetros permiten:
  - Cuantizar la entrada X en tiempo real
  - Reconstruir aproximadamente Y_f para la siguiente capa

B. Calibration

1) Proceso de calibración:

- Se realiza inferencia usando entradas de ejemplo
- Se observan los valores típicos de salida

2) Obtención de parámetros:

- Se determinan valores razonables para:
  - α (límite superior)
  - β (límite inferior)
- Con estos se calculan:
  - El escalador s
  - El valor z (zero point)

3) Contexto de aplicación:

- Este proceso se realiza durante la etapa de:
  - Post-training quantization

4) Fórmulas relacionadas:

$$x_f = s(x_q - z)$$

$$x_f = sx_q$$

C. Relación entre procesos

- Dynamic Quantization: Transforma a enteros la entrada de la capa (X)
- Calibration: Transforma a flotantes la salida de la capa que está en enteros (Y_q)

VII. Otras estrategias

A. MSE (Error Cuadrático Medio)

- Selecciona $[\beta, \alpha]$ para minimizar el error entre el vector original y el cuantizado
- Implementación mediante GridSearch para encontrar los óptimos β, α

B. Cross-Entropy

- Utilizado cuando los valores del tensor tienen importancia desigual
  - Softmax Layer LLM
  - Greedy
  - Top-P

VIII. Quantization Granularity

A. Aplicación en capas convolucionales

- Las convoluciones contienen múltiples filtros
- Cada filtro:
  - Tiene valores y distribuciones distintos
  - Detecta diferentes características en la imagen

B. Enfoque de quantization

- No se puede usar el mismo $[\beta, \alpha]$ para todos los filtros
- Solución:
  - Calcular valores individuales de β, α por filtro
  - Mayor precisión al preservar las distribuciones específicas

IX. Post Training Quantization (PTQ) y Quantization Aware Tuning (QAT)

A. Post Training Quantization

El PTQ utiliza datos no vistos durante el entrenamiento para que observadores (observers) especializados calculen parámetros estadísticos específicos para cada capa del modelo. Este enfoque permite adaptar la cuantización a las características reales de los datos que procesará el modelo en producción.

B. Quantization Aware Tuning

El QAT, por su parte, incorpora módulos que simulan los efectos de la cuantización durante la fase de entrenamiento. Al incluir estos efectos en la función de pérdida, el modelo aprende a compensarlos, resultando en modelos más robustos y con mejor rendimiento bajo esquemas de cuantización.

X. Unsupervised Learning vs Supervised Learning

A. Supervised Learning

- Vector de características:

$$X = \{x_1, x_2, ..., x_D\}$$

- Conjunto de datos completo:

$$\{X_i, y_i\}_i^N$$

donde:

  - X_i: Features/atributos
  - y_i: Etiquetas (pueden ser):
    - Números (regresión)
    - Clases (clasificación)
    - Otros tipos de etiquetas

B. Unsupervised Learning

- Solo contiene vectores de features:

$$\{X_i\}_i^N$$

- No existen etiquetas y_i
- Objetivo principal:
  - Transformar las features para resolver problemas

C. Aplicaciones principales

1) Reducción de dimensionalidad:

- Técnica principal: Autoencoder
- Reduce el número de features manteniendo la información clave

2) Detección de outliers:

- Problema binario (normal/anómalo)
- Algoritmo principal: DBSCAN

XI. Clustering

El clustering consiste en identificar grupos naturales dentro de los datos basándose en la similitud de sus características. Este proceso divide el espacio de características en regiones homogéneas, aunque existen múltiples interpretaciones de lo que constituye un "grupo". Las técnicas de clustering pueden clasificarse en tres categorías principales: jerárquicas (que construyen sistemáticamente grupos basados en métricas de similitud), determinísticas (donde cada muestra pertenece claramente a un cluster) y probabilísticas (que asignan probabilidades de pertenencia, como en el caso de Gaussian Mixture Models).

A. K-Means

El algoritmo K-Means representa un enfoque determinístico que asigna todas las muestras a clusters específicos, minimizando una función de costo basada en el error cuadrático. Mediante un proceso iterativo, el algoritmo ajusta los centroides hasta converger a una solución local óptima, proporcionando una partición clara del espacio de características.