Escuela Politécnica Nacional

Nombre: Francisco Ulloa
Fecha: Quito, 28 de enero de 2026
Tema: Factoreo en transformers
Repositorio:
https://github.com/Fu5CHAR/Metodos_numericos_2025B_Ulloa-Francisco/tree/main

Uso del factoreo de matrices en arquitecturas Transformer

¿Qué es el factoreo de matrices en Transformers?

En redes neuronales **Transformer**, el factoreo de matrices consiste en **descomponer una matriz grande en el producto de matrices más pequeñas**, normalmente de bajo rango:

$$W \in \mathbb{R}^{d \times d} \Rightarrow W \approx A \cdot B$$

donde:

- $A \in \mathbb{R}^{d \times k}$
- $B \in \mathbb{R}^{k \times d}$
- $conk \ll d$

Este enfoque aparece explícita o implícitamente en:

- Low-Rank Factorization
- Attention factorization
- Proyecciones Q, K y V
- Adapter layers
- LoRA (Low-Rank Adaptation)
- Transformers eficientes (Linformer, Performer, etc.)

¿Dónde se usa el factoreo de matrices en Transformers?

1. Factoreo en el mecanismo de atención

La atención estándar se define como:

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

Este cálculo tiene complejidad **cuadrática**

$$O(n^2)$$

respecto a la longitud de la secuencia.

Mediante factoreo o proyecciones de bajo rango:

- Se reduce el costo computacional
- Se reduce el consumo de memoria
- Se habilita el uso de secuencias más largas

Ejemplo: **Linformer** aproxima la matriz de atención usando proyecciones de bajo rango.

---

2. Factoreo en capas lineales (Feed-Forward)

Las capas Feed-Forward contienen matrices grandes:

$$W \in \mathbb{R}^{d_{model} \times d_{ff}}$$

Estas matrices pueden factorizarse como:

$$W = A \cdot B$$

Esto se usa para:

- Compresión del modelo
- Reducción de parámetros
- Aceleración de inferencia

---

3. Factoreo en fine-tuning eficiente (LoRA)

En **LoRA**, los pesos originales del modelo se congelan y se añaden matrices de bajo rango entrenables:

$$W' = W + A \cdot B$$

Ventajas:

- Se entrenan pocos parámetros
- No se altera el modelo base
- Permite adaptar modelos grandes de forma eficiente

Razones para usar factoreo de matrices en Transformers

1. Reducción del costo computacional

- Las multiplicaciones matriciales grandes son costosas

- El factoreo reduce la complejidad de:

$$O(d^2) \rightarrow O(d \cdot k)$$

---

2. Ahorro de memoria

- Menor número de parámetros
- Menor uso de memoria de activaciones
- Clave para entrenamiento en GPUs

---

3. Escalabilidad del modelo

- Permite:
  - Modelos más grandes
  - Secuencias más largas
  - Mayor profundidad
- Sin crecimiento explosivo del costo computacional

---

4. Regularización implícita

- El bajo rango actúa como restricción
- Reduce el sobreajuste
- Mejora la generalización

---

5. Eficiencia en fine-tuning

- Se entrenan solo matrices pequeñas
- Reduce tiempo y costo de entrenamiento
- Ideal para personalización de LLMs

---

6. Mejor aprovechamiento del hardware

- GPUs modernas están optimizadas para multiplicaciones matriciales pequeñas
- Mejor uso de Tensor Cores (FP16, FP8)

Ventajas prácticas del factoreo de matrices

Ventaja	Impacto
Menos parámetros	Modelos más ligeros
Menor latencia	Inferencia más rápida
Menor consumo energético	Mayor eficiencia

Ventaja	Impacto
Mayor escalabilidad	Secuencias largas
Fine-tuning económico	Menor costo
Regularización	Mejor generalización

Conclusión

El factoreo de matrices es una **estrategia clave** en los Transformers modernos que permite:

- Reducir costos computacionales
- Escalar modelos de lenguaje grandes
- Adaptar modelos sin reentrenamiento completo
- Aprovechar hardware especializado

Sin factoreo de matrices, los Transformers modernos no serían viables a gran escala.