Architetture e Programmazione dei Sistemi di Elaborazione
Progetto a.a. 2022/23

"Attention mechanism"
in linguaggio assembly x86-32+SSE, x86-64+AVX e openMP

Fabrizio Angiulli Fabio Fassetti Simona Nisticò

1 Decrizione del problema

Nel campo dell'intelligenza artificiale, le reti neurali svolgono attualmente un ruolo centrale, questo grazie ai notevoli risultati che hanno conseguito nell'automatizzare una serie di compiti che in genere venivano svolti da agenti umani. Le tipologie di architetture basate su tale paradigma sono molteplici e tra queste hanno conquistato particolare popolarità quelle che includono al loro interno il *meccanismo di attenzione*.
Il meccanismo di attenzione consente alla rete neurale di concentrarsi maggiormente sulle porzioni potenzialmente più significative dell'input, riducendo quindi l'impatto del rumore che potrebbe essere presente all'interno dei dati e migliorando così le prestazioni.

Le reti neurali

Una rete neurale è costituita da una complessa rete di connessioni tra neuroni che sono in grado, cooperando tra essi, di realizzare un compito specifico; ciascun neurone rappresenta un elemento computazionale semplice ed il loro insieme viene organizzato in una struttura che abbia un sufficiente grado di parallelismo.
In genere, un neurone di una rete neurale artificiale riceve molti ingressi, a cui viene associato un peso in corrispondenza di ciascuno di essi, e dà in uscita un singolo valore rappresentato da una funzione della somma pesata degli ingressi.
L'elaborazione eseguita da ciascun neurone deve essere necessariamente locale, ovvero il suo output deve dipendere esclusivamente dall'ingresso corrente e da ciò che è contenuto nella sua memoria locale.
Un neurone ha n canali di ingresso x_i con $i \in 1, \ldots, n$, ciascuno associato ad un peso w_i, rappresentato da un numero reale. Il segnale in uscita viene ricavato mediante la funzione di attivazione, valutata in corrispondenza della somma pesata degli ingressi a cui viene aggiunto un ulteriore ingresso che prende il nome di bias, b come mostrato in Figura 1.
Questo, in termini matematici, si traduce nella seguente formulazione:

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right)$$

Una rete neurale è costituita da più strati di neuroni, detti livelli, disposti in maniera tale che ciascuno di essi sia connesso con quelli appartenenti allo strato successivo e che non ci sia alcuna

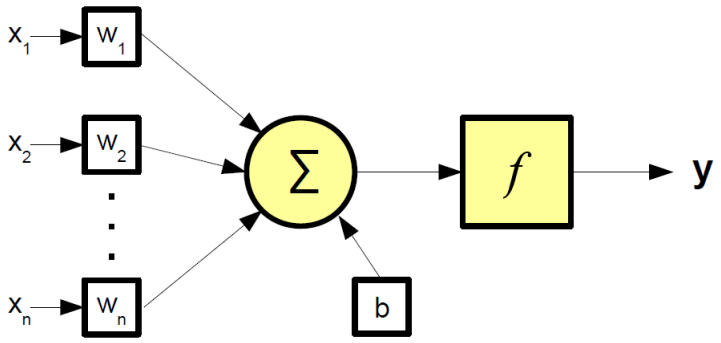

Figura 1: Struttura del neurone

comunicazione tra neuroni dello stesso livello. L'input della rete neurale viene fornito ai neuroni del primo livello, i livelli intermedi sono generalmente "nascosti", ovvero non comunicano in maniera diretta con l'esterno, e l'output della rete viene prodotto dall'ultimo livello, come mostrato in Figura 2.

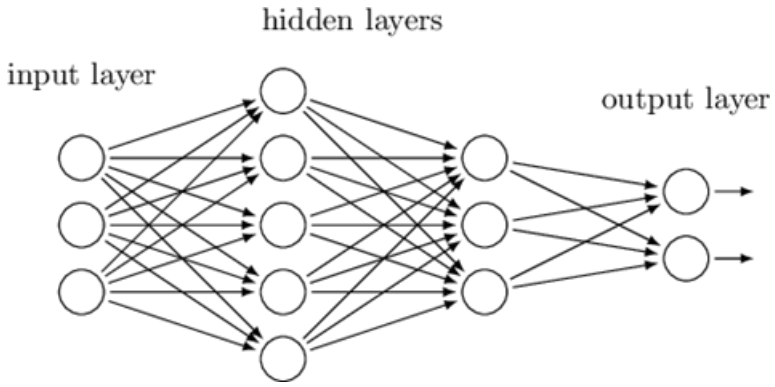

Figura 2: Esempio di struttura di una rete neurale

L'insieme dei livelli e la loro struttura organizzativa prende il nome di *architettura della rete*.

Architettura di riferimento

Una tra le architetture di classificazione più diffuse che sfrutta il meccanismo di attenzione è riportata in Figura 3.
In generale, un'architettura di classificazione riceve in input dei dati e restituisce, per ognuno di essi, la classe più probabile di appartenenza.
L'architettura consta di tre componenti: l'*encoder*, il *modulo di attenzione* ed il *classificatore*.
L'encoder riceve in ingresso i dati di input e sintetizza le informazioni trasformandolo ogni dato in un matrice di feature che sia efficacemente elaborabile da parte dei componenti successivi.
Il modulo di attenzione riceve in ingresso i dati prodotti dall'encoder e li trasforma in maniera tale che le features più rilevanti acquistino maggiore importanza.
Il classificatore riceve in ingresso l'output del modulo di attenzione e predice, per ogni dato, una classe di appartenenza.

Figura 3: Esempio di architettura per il task di classificazione che utilizzi il meccanismo di attenzione

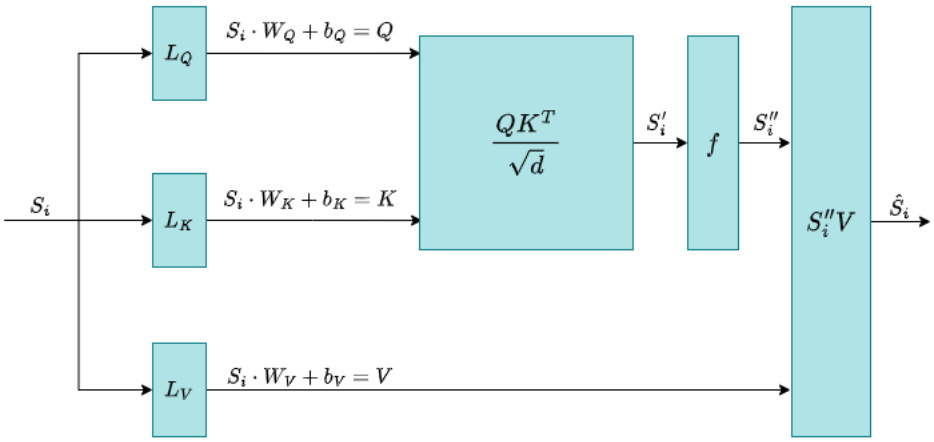

Figura 4: Struttura dettagliata del meccanismo di attenzione.

Il meccanismo di attenzione

Il meccanismo di attenzione (*Attention Mechanism*) è molto noto nel campo delle reti neurali in quanto ha consentito di ottenere un miglioramento nelle performance raggiunte in una serie di task. Questo fu originariamente introdotto per migliorare le performance dei modelli di traduzione automatica ed è oggi impiegato anche al di fuori dei dati testuali, ad esempio su immagini ed audio. L'idea di base è quella di consentire al decoder di focalizzarsi dinamicamente sulle parti più importanti dell'input dando a queste ultime un peso maggiore. Quello che si ottiene è una combinazione pesata dei vettori di input dove quelli più rilevanti avranno un peso maggiore rispetto agli altri.
Nella Figura 4 è riportata in dettaglio la struttura del modulo di attenzione. L'input x di dimensione $n \times d$ viene dato parallelamente in ingresso a tre livelli, L_Q, L_K ed L_V composti da n_n neuroni, i quali producono in output le matrici Q, K e V. Successivamente, viene calcolato il prodotto matriciale $Q \times K^T$, scalato per un fattore pari a $\sqrt{d}$. Alla matrice così ottenuta viene applicata un'opportuna funzione di attivazione f, il risultato viene moltiplicato per la matrice V e fornito in output.
Per incrementare il livello di parallelismo, i dati di input x vengono dati in ingresso al meccanismo di attenzione a gruppi di s elementi.

Preliminari

Un tensore S, nel contesto in esame, è una struttura tre-dimensionale $s \times n \times d$, dove:

- $S_i, i \in \{1..s\}$ denota l'i-esima matrice bi-dimensionale $n \times d$ in S;

- $S_{i,j}, j \in \{1..n\}$ denota la j-esima riga di S_i;

Dato un tensore S, il trasposto di S, S^T, è un tensore S' tale che

$$S'_i = S_i^T$$

Dato un tensore S, la somma tra un vettore h di d elementi ed S è un tensore S' tale che

$$S'_{i,j} = S_{i,j} + h.$$

Dato un tensore S, il prodotto tra una matrice H bi-dimensionale $d \times k$ ed S è un tensore S' tale che

$$S'_i = S_i \cdot H$$

dove $\cdot$ rappresenta il prodotto matriciale.
Dati due tensori $S1$, $S2$, il prodotto tra $S1$ ed $S2$ è un tensore S' tale che:

$$S'_i = S1_i \cdot S2_i$$

dove $\cdot$ rappresenta il prodotto matriciale.
Un *dataset* è un insieme di N tensori, ossia matrici di dimensione $s \times n \times d$.
Data una rete neurale e dato un livello denso D in tale rete, a questo sono associati una matrice di pesi W_D, di dimensione $d \times n_n$, e un vettore di bias b_D composto da n_n elementi. Dato un tensore x, l'output di D è dato dal risultato della seguente combinazione lineare:

$$x \cdot W_D + b_D.$$

Attention Mechanism

Come già accennato nell'introduzione, il meccanismo di attenzione A rappresenta una porzione della rete neurale ed è composto da tre livelli densi, D_Q, D_K e D_V. A riceve un input S, proveniente dai livelli precedenti della rete. S viene dato in ingresso ai tre livelli densi in A, il cui output viene combinato e viene fornito in output una versione $\hat{S}$ dell'input dove le componenti sono sintetizzate in accordo a quanto previsto dal meccanismo di attenzione come descritto sopra. Dato l'i-esimo elemento S_i di S, avente dimensione $n \times d$, il meccanismo di attenzione viene implementato seguendo quattro fasi

1. Per ogni livello denso viene calcolato l'output

$$Q = S_i \cdot W_Q + b_Q \tag{1}$$
$$K = S_i \cdot W_K + b_K \tag{2}$$
$$V = S_i \cdot W_V + b_V \tag{3}$$

2. L'output dei primi due livelli viene combinato in accordo a

$$\frac{Q \cdot K^T}{\sqrt{d}} \tag{4}$$

3. Al risultato della seconda fase viene applicata la funzione

$$f(x) = s(x)\left(\frac{-1}{x+2} + 0.5\right) + 0.5 \quad \text{con } s(x) = \begin{cases} 1, & \text{se } x >= 0 \\ -1, & \text{se } x > 0 \end{cases} \tag{5}$$

4. La matrice calcolata nella terza fase viene moltiplicata per la matrice V e restituita come output del maccanismo di attenzione $\hat{x}$, quindi, complessivamente

$$\hat{x} = f\left(\frac{Q \cdot K^T}{\sqrt{d}}\right) \cdot V \tag{6}$$

Algoritmo

L'algoritmo 1 riassume lo pseudocodice della metodologia.

ALGORITMO 1: Attention mechanism

Input: un dataset DS di dimensione $N \times s \times n \times d$, le matrici dei pesi W_Q, W_K, W_V, i vettori di bias b_Q, b_K, b_V

Output: un dataset $\hat{DS}$

1 **begin**
2 **foreach** *tensore* $S \in DS$ **do**
3 $Q = S \cdot W_Q + b_Q$
4 $K = S \cdot W_K + b_K$
5 $V = S \cdot W_V + b_V$
6 $S' = \frac{Q \cdot K^T}{\sqrt{d}}$
   `// La divisione di un tensore per uno scalare si effettua dividendo tutti gli elementi del tensore per lo scalare`
7 $S'' = f(S')$
   `// Dove la funzione` f `è quella definita nell'Equazione` (5)
8 $\hat{S} = S'' \cdot V$
9 **return** $\hat{DS}$ composto da N tensori $\hat{S}$

2 Descrizione dell'attività progettuale.

Obiettivo del progetto è mettere a punto un'implementazione dell'algoritmo Attention mechanism in linguaggio C e di migliorarne le prestazioni utilizzando le tecniche di ottimizzazione basate sull'organizzazione dell'hardware.
L'ambiente sw/hw di riferimento è costituito dal linguaggio di programmazione C (`gcc`), dal linguaggio assembly x86-32+SSE e dalla sua estensione x86-32+AVX (`nasm`) e dal sistema operativo Linux (`ubuntu`).
In particolare il codice deve consentire di trovare il dataset trasformato $\hat{DS}$ con l'algoritmo 1, dati i valori dei parametri: `-si <`si `prima dimensione del tensore>` e `-n <`n `seconda dimensione del tensore>`.
Quindi la chiamata avrà la seguente struttura:

```
./att<arch> -ds <DS> -wq <WQ> -wk <WK> -wv <WV> -bq <bQ> -bk <bK> -bv <bV> -si <si> -n <n>
    <arch>: architettura associata all'eseguibile, {32c, 64c, 32ompc 64ompc}
    <DS>: file ds2 contenente il dataset
    <WQ>: file ds2 contenente i pesi WQ
    <WK>: file ds2 contenente i pesi WK
    <WV>: file ds2 contenente i pesi WV
    <bQ>: file ds2 contenente i pesi bQ
    <bK>: file ds2 contenente i pesi bK
    <bV>: file ds2 contenente i pesi bV
    <si>: prima dimensione del tensore
    <n>: seconda dimensione del tensore
```

e sorgenti ed eseguibili devono essere contenuti in una cartella il cui nome è l'id del gruppo. Qualora un valore di un parametro (sia esso di default o specificato dall'utente) non sia applicabile, il codice deve segnalarlo con un messaggio e terminare.

Di seguito si riportano ulteriori linee guida per lo svolgimento del progetto:

- Si consiglia di affrontare il progetto nel seguente modo:

  1. Codificare l'algoritmo interamente in linguaggio C, possibilmente come sequenza di chiamate a funzioni;
  2. Sostituire le funzioni scritte in linguaggio ad alto livello che necessitano di essere ottimizzate con corrispondenti funzioni scritte in linguaggo assembly.

  Ciò consentirà di verificare che l'algoritmo che si intende ottimizzare è corretto e di gestire più facilmente la complessità del progetto.

- Al fine di migliorare la valutazione dell'attività progettuale è preferibile presentare nella relazione un confronto tra le prestazioni delle versioni intermedie, ognuna delle quali introduce una particolare ottimizzazione, e finale del codice. Obiettivo del confronto è sostanziare la bontà delle ottimizzazioni effettuate.

- Occorre lavorare in autonomia e non collaborare con gli altri gruppi. Soluzioni troppo simili riceveranno una valutazione negativa. I progetti verranno messi in competizione.

- Sono richieste due soluzioni software, la prima per l'architettura x86-32+SSE e la seconda per l'architettura x86-64+AVX.

  Per i dettagli riguardanti la redazione del codice fare riferimento al file sorgente *c* `att32c.c`, al file sorgente *nasm* `att32.nasm`, allo script eseguibile `runatt32` (versione `x86-32+SSE`) e al file sorgente *c* `att64c.c`, al sorgente *nasm* `att64.nasm`, allo script eseguibile `runatt64` per la versione `x86-64+AVX` disponibili sulla piattaforma.

  Per l'interfacciamento tra programmi in linguaggio C e programmi in linguaggio assembly fare riferimento al documento allegato alla descrizione del progetto.

- È inoltre richiesta per ogni soluzione, una versione che faccia uso delle istruzioni `OpenMP`. I nomi dei relativi file dovranno contenere il suffisso "`_omp`" (es. `att32_omp.c`).

- Il software dovrà essere corredato da una relazione. Per la presentazione del progetto è possibile avvalersi di slide.

- Prima della data di consegna del progetto verranno pubblicate le convenzioni da rispettare riguardanti i nomi e la collocazione dei file/directory al fine della compilazione e l'esecuzione dei codici di programma mediante script appositamente predisposti. **Dato l'elevato numero di progetti, sarà cura del candidato accertarsi di aver rispettato pienamente le convenzioni di consegna.**

Buon lavoro!