Protocollo Genetico

Una architettura vettoriale binaria frattale per AI – Dario Limatola

Spazio vettoriale

Sia dato uno spazio vettoriale discreto in Z, con Z dimensioni e Z metadimensioni (tensori).

Il sistema possiede un punto costante, detto **punto 0**, che rappresenta l'origine computazionale assoluta:

v0:=T(0,0,...,0)

dove

- v0 è immutabile
- Nessuna mutazione può alterarlo
- Ogni vettore deve poterne accedere e ancorarsi ad esso
- Rappresenta la radice logica e cognitiva dell'intero sistema

Utilizzeremo per gran parte delle nostre operazioni il sottoinsieme di N di Z.

Gene

Chiamiamo “Gene” una struttura a pila FIFO comprendente:

- Un vettore di booleani di dimensione N: vi=[bi1,bi2,...,bin]∈{0,1}
- Un set di N istruzioni
- Una macchina di Turing che elabora dati e istruzioni

D’ora in poi li chiameremo semplicemente “vettori” o “dimensioni”.

Avremo quindi un vettore binario, chiamato Gene, con determinate proprietà:

- È una struttura binaria, deterministica e discreta
- È pienamente Turing-capable
- È estremamente leggero e comprimibile in un unico vettore binario
- È ricorsivo: ogni istruzione sul vettore x può essere a sua volta un insieme di sotto istruzioni più specifiche in altre dimensioni o tensori
- E' virtualizzabile, lasciando le operazioni matematiche fuori dalla sandbox
- È evolutivo tramite il Dilemma
- È ricorsivo: possiamo creare nuove dimensioni e tensori
- E' una struttura dati per agenti intelligenti modulari: Ogni agente può avere la sua Matrice genetica
- Può essere associato ad un transformer o una struttura dati, a grafo ad esempio, dove i nodi sono rappresentati con token o indirizzi e i vertici vettori a più dimensioni, relazionali, possono rappresentare spazi di memoria e le relazioni tra di essi

Regole di base

Abbiamo delle regole e delle operazioni che possiamo compiere su tali vettori: quelle su pile, quelle algebriche e operazioni vettoriali discrete. Possiamo creare nuove dimensioni e tensori, ricorsivamente. La struttura a pila assicura l'integrità dei dati. Operazioni più complesse, come riordinamento dei vettori, modifica radicale, aggiornamento dell'ordine dei vettori, operazioni di taglia e incolla informazioni) sono compiute tramite permutazioni lungo tutto l'insieme Z.

Un vettore o un tensore lungo tutto l'insieme Z, è chiamato Dilemma: avrà due versioni differenti dello stesso vettore, quella di base in N e quella modificata in -N. la versione migliore sarà scelta di volta in volta secondo feedback esterno, algoritmi di consenso o di ottimizzazione.

ESEMPIO DI FUNZIONAMENTO

Istruzioni fisse:

Vettore in N di lunghezza 5:

[0] = "Salva" [1] = "Cancella" [2] = "Analizza" [3] = "Rispondi" [4] = "Pondera"

Ogni istruzione è a sua volta un sotto vettore.

Stato iniziale

Pile: ["1", "1", "1",”0”,”0”]
Istruzioni: ["Salva", "Cancella", "Analizza"]

Operazione: ricevi feedback positivo → `PUSH(3, "1")`

Pile aggiornata: ["1", "1", "1", "1",”0”]
Istruzioni: ["Salva", "Cancella", "Analizza","Rispondi"]

Operazione non FIFO: vuoi correggere "Cancella" in "Pondera"

Questo è il Dilemma:

Non puoi modificare direttamente, generi una matrice modificata in -N:

-Matrice: ["0", "0", "1", "1",”1”]

Istruzioni: ["Analizza", "Rispondi","Pondera",]

Viene eseguito un confronto. Se validato, la nuova matrice diventa ufficiale e viene permutata.

Dimensioni

Struttura a 2 dimensioni: Mij∈{0,1}

- Colonne → istruzioni fisse:
  - Ogni colonna rappresenta un'istruzione semantica invariabile **I**

- Righe → istanze, o entità:
  - Ogni **riga i** è una **pila temporale o spaziale** di istruzioni, riferita a un determinato **agente** o identità computazionale **V**

	I_1 (es. Salva)	I_2 (Cancella)	I_3 (Decidi)	...
v_1	1	0	1	...
v_2	0	0	1	...
v_3	1	1	0	...

Questa struttura può qui rappresentare il coordinamento tra diversi processori o threads.

A sua volta ogni istruzione avrà vettori di microistruzioni.

Al crescere delle dimensioni e dei tensori crescono anche gli usi possibili:

- Possiamo avere vettori temporali, che possono essere settati sul giorno o il secondo, ed essere connessi tramite un'unica matrice ad ogni operazione effettuata in quel giorno.
- Possiamo introdurre una maggiore granularità dell'informazione: alla semplice istruzione binaria possiamo associare "vettori bucket" dove un dato binario viene associato a un dato più specifico: possiamo avere diversi bucket, per esempio possiamo tracciare in diverse dimensioni se una data informazione (per esempio un confidence score o una probabilità statistica) appartiene a sua volta a bucket ricorsivi di base 2 o 10 (per visualizzare 100 buckets avremo bisogno di 10 ricorsivo ($10\text{^}1$ Ciò ci permette una maggiore e sempre crescente approssimazione sempre operando su dati binari.
- Può rappresentare un qbit con i suoi valori 1,0,-1 e le sue superposizioni in diverse dimensioni.

- Possiamo abbinarlo a un transformer e creare una LLM: ogni token può essere associato a varie dimensioni e tensori quali la lingua, il contesto storico, il tono, il contenuto emotivo, la semantica.
  L'informazione binaria è meno dettagliata, ma al crescere delle dimensioni e dei tensori e dei collegamenti può rappresentare strutture estremamente complesse di rimandi e collegamenti, in maniera estremamente efficiente e deterministica e rimuovendo il rumore di fondo.

Ricorsività

Ogni tensore e metatensore può introdurre una maggiore complessità: in una LLM il primo tensore analizzerà caratteri, il secondo le parole, il terzo connessioni di frasi e pensieri. A loro volta gli strati superiori saranno collegati a quelli inferiori tramite vettori.

La generalizzazione frattale è un tensore:

$$T: D_1 \times D_2 \times \cdots \times D_k \rightarrow B^*$$

dove ogni dimensione Dj rappresenta un contesto (tempo, livello, spazio, logica). Ogni punto del tensore contiene una pila, e ogni pila può contenere un **metadato** $M \subseteq B^*$ con proprietà autoespandibili.

La differenziazione tra dati e metadati è qui annullata, avremo livelli più bassi (i token) e livelli a maggiore astrazione, in strutture ricorsive e dinamiche.

Implementazione

Una LLM con uno spazio vettoriale di token, in diverse lingue, in diverse dimensioni (regole grammaticali, semantiche, logiche, lingua, contesto linguistico etc.).

Un parser.

Tensori relativi a dimensioni, lingua, grammatica, logica etc ad esso collegati. Avremo una serie di vettori tra le dimensioni del 1 livello e quelli del seondo livello. Potremo associare un token quindi ad una lingua, ad un contesto, e a una fitta serie di vettori binari.

Metatensori che analizzano la struttura delle frasi, con regole e ragionamenti, inferenze.

Via via, sempre maggiori livelli di astrazione espressi come metatensori.

Tutte Geni, ovvero pile di vettori binari associati a un set di istruzioni binarie capaci di esprimere una macchina di Turing.

La granularità sarà associata a determinati vettori “bucket”, che possono esprimere ad esempio valori da 1 a 100 ognuno con specifici vettori. Un byte ad esempio sarà rappresentato con 2^3 buckets, ovvero ogni valore binario avrà una precisione crescente, con una divisione di un singolo dato binario in 2, tre volte ricorsivamente (lungo lo stesso vettore).

Data la natura binaria deterministica e discreta della nostra struttura ogni token è collegato solo a determinati vettori, ciò consente una maggiore efficienza nella gestione dei dati, e, unitamente alla semplicità delle operazioni, può essere più energeticamente efficiente di una LLM tradizionale di diversi ordini di grandezza.

Ogni parametro è a 1 bit, ciò ci permette un maggiore numero e complessità di parametri, la cui complessità e precisione sarà sempre un’approssimazione discreta su molteplici dimensioni vettoriali.

Stuttura a grafo, capace di espandersi, evolversi e migliorare nel tempo, con una memoria estremamente efficiente consistente in una mappa di token e di frasi unite tramite vettori binari.

Tranformer esterno (open point) .