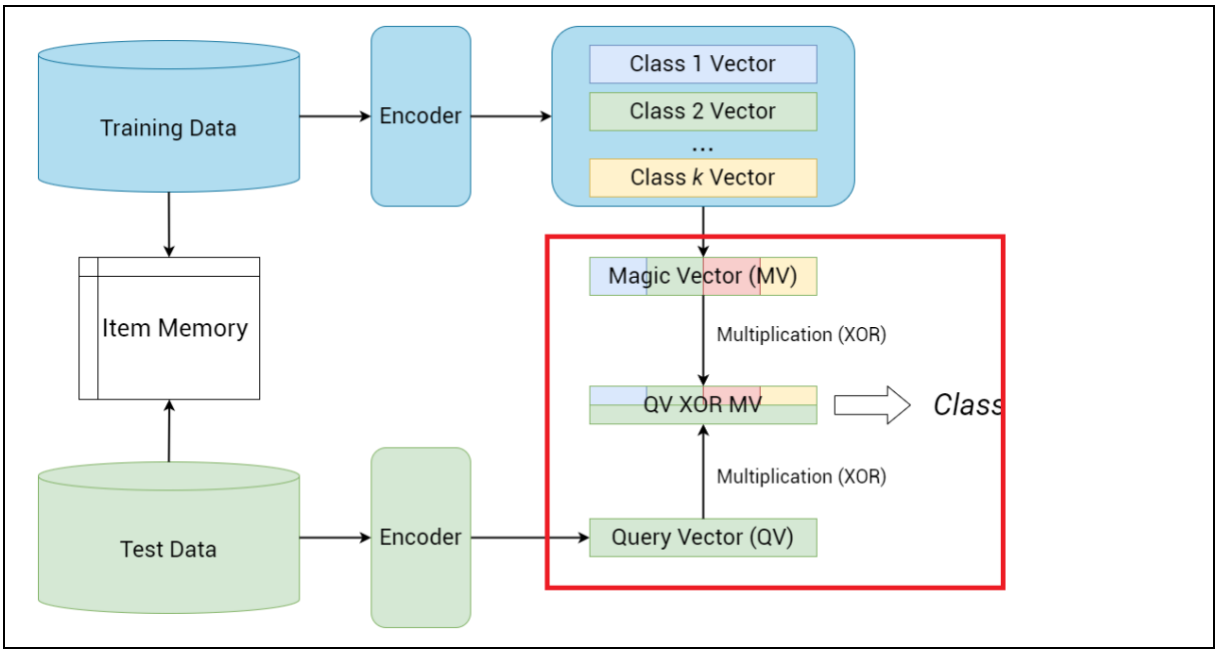
Un nuovo algoritmo di ricerca di somiglianza in un singolo passaggio nella memoria associativa per il calcolo iperdimensionale

Luigi Loreti, Giovanni Todeschini, Francesco Regazzoni, Alessio Fumagalli, Luca Muscarnera

Rebel Dynamics srl + MOC Polimi

Astratto

L'Hyperdimensional Computing (HDC) è un framework di calcolo emergente, con promettenti applicazioni per l'AI e ambienti correlati (Machine Learning, Deep Learning, Hopfield network, etc.).
La nostra azienda (OPTOPC SA) sta pionieristicamente sviluppando l'HDC su un innovativo hardware ottico brevettato, sfruttando al massimo il suo parallelismo intrinseco e la semplicità delle operazioni bit-to-bit per accelerare significativamente il calcolo parallelo ottico (OPC).
*Uno dei punti cruciali dell'HDC è l'**inferenza** o la ricerca della somiglianza del vettore di query con i vettori memorizzati nella Memoria Associativa (AM), che richiede migliaia di operazioni a seconda della dimensione dell'Ipervettore (HV) e del numero di vettori memorizzati nell'AM.*
*È noto che una ricerca in un database non ordinato richiede in media **n**/2 tentativi, dove **n** è il numero di Item memorizzati nella AM. Diversi approcci sono stati esplorati per risolvere il problema, fino al famoso algoritmo quantistico di Grover che, pur distinguendosi per la sua soluzione della ricerca in tempo che scala come la radice di n, rappresenta un notevole miglioramento rispetto ai metodi tradizionali.*
Grazie al nostro algoritmo, siamo in grado di risolvere questa ricerca in un solo passaggio computazionale indipendentemente dal numero di elementi in memoria e al numero di dimensioni dell'HV, riducendo drasticamente i tempi di calcolo e il consumo energetico.
*Questo algoritmo permette anche di risolvere l'altro punto fondamentale dell'HDC che è la **fattorizzazione**, ovvero trovare la componente di una moltiplicazione di due o più HV dell'Item Memory, detta anche Memoria di pulizia. Su quest'ultimo tema troviamo in letteratura molti metodi come la Resonant Network, ma tutti richiedono molti passaggi e quantità di calcolo. Con l'algoritmo proposto viene risolto in uno solo o due passaggi per 2 elementi e in n/2 passaggi per 3 elementi.migliorando le prestazioni di una rete di risonatori con un algoritmo migliorato ed eliminazione delle ricerche di somiglianza nei codebook.*
Questi algoritmi si prestano ad essere integrati in diver.se architetture computazionali, non solo per il calcolo ottico, ma anche per il calcolo in memoria (PCM, FPGA, ASIC, ecc.).
Riteniamo che il nostro innovativo approccio aumenterà la potenza di calcolo dell'HDC.



Flusso dell'algoritmo (in rosso)

Descrizione HDC

L'Hyperdimensional Computing (HDC) è un framework di calcolo emergente, ispirato al cervello umano, dove la enorme presenza di neuroni e delle loro connessioni ha ispirato un nuovo paradigma di calcolo per simulare i comportamenti intelligenti. Si basa su un concetto di spazio multidimensionale, tipicamente da 1000 a 10000 dimensioni, in cui vengono definiti i suoi componenti come vettori ad alte dimensioni, detti ipervettori (*hypervectors*, HV) le cui componenti possono essere binarie, bipolari, intere o complesse.
In uno spazio ad alte dimensioni una delle caratteristiche fondamentali è quella che é possibile costruire insiemi di vettori di grande cardinalitá e con un basso grado di correlazione reciproca, misurabile attraverso l'operazione algebrica del prodotto scalare Questa proprietà prende il nome di a "quasi-ortogonalità", e il gran numero di possibili realizzazioni di vettori quasi ortogonali risulta coerente con il risultato teorico presentato da Kainen e Kurkova [Kainen & Kurkova] in cui viene esposta la crescita esponenziale del numero di vettori quasi ortogonali in funzione della dimensione dello spazio, in contrapposizione alla crescita lineare del numero di vettori ortogonali in senso classico.

Questa proprietà viene utilizzata per associare HV casuali a concetti elementari, quali forme, colori, lettere dell'alfabeto, posizioni, immagini in modo tale che non ci sia correlazione tra i vari oggetti. ~~definiti~~, Questi oggetti vengono utilizzati come elementi atomici di concetti, simboli e vengono memorizzati in una memoria chiamata memoria degli elementi (*item memory*, IM). Con delle operazioni matematiche molto semplici questi elementi atomici possono essere uniti, legati, associati, sequenziati per formare idee, concetti, grafi, alberi, sequenze temporali e spaziali, in definitiva delle classi di comportamenti, classi di proprietà, categorie o percorsi. Queste classi sono rappresentate da HV come risultato delle operazioni fatte tra i simboli atomici e vengono memorizzati, in fase di addestramento, in una memoria detta Memoria associative (*associative memory*, AM). L'addestramento consiste nel raggruppare concetti simili, comportamenti simili, immagini o testi simili nelle classi di appartenenza. Quindi si può dire che la AM memorizza diverse classi di immagini, oggetti composti, testi, comportamenti definiti dalle codifiche fatte tramite operazioni matematiche.
Le informazioni esaustive sul HDC possono essere recuperate su kanerva2022hdmss.pdf (berkeley.edu), qui daremo solo le principali proprietà.
Le proprietá geometriche dei vettori ad alte dimensioni, sono in una certa misura indipendenenti dalle proprietá dello spazio matematico di riferimento e seguono – a livello generale – le nozioni imposte dalla teoria della concentrazione della misura [Vershynin https://www.math.uci.edu/~rvershyn/papers/HDP-book/HDP-book.pdf]. Alla luce di questa universalitá nel comportamento di oggetti geometrici in dimensione elevata incentreremo la discussione successiva sulla geometria dei vettori bipolari, ovvero composti da solo due possibili elementi in maniera coerente con applicazioni per calcolo alte prestazioni computazionali

Tutte le operazioni di manipolazione di HV si riducono a quattro operazioni fondamentali:

1) *Misura della distanza o somiglianza* tra due vettori, che per gli HV bipolari o binari è rappresentata dalla distanza di Hamming, cioè dal numero di bit differenti nella stessa posizione tra due HV,
2) Somma (+) bit per bit di due o più HV, che rappresenta l'associazione o sovrapposizione di elementi che rappresentano una categoria, per esempio **forme** = **quadrato** + **rettangolo** + **cerchio**... oppure **colori**=**rosso** + **verde** +**blu**+...In questo modo si ottengono due classi determinate dagli ipervettori **forme** e **colori.** La somma va normalizzata a maggioranza di 0 o di 1 per riportarla ad un HV binari ed è simile ai suoi componenti, ossia: Se gli 0 sono la maggioranza nel dataset, il dato (o il segnale) viene normalizzato a **0**. Se gli 1 sono la maggioranza, il dato viene normalizzato a **1**. Se il numero di 0 e 1 è uguale, può essere adottata una regola di arbitraggio per cui si normalizza ad 1 (o, equivalentemente, a 0).
3) *Prodotto* (XOR o *) di due o più HV bit per bit, che rappresenta il legame tra due concetti atomici, per esempio **QR**= **forme** * **quadrato** + **colore***r**osso**, determinando un nuovo oggetto che ha come forma un quadrato e come colore il rosso. Il prodotto genera un HV che è dissimile dai suoi componenti.
4) *Permutazione (P o shift 0 <),* ottenuta come rotazione delle posizioni dei bit di HV, serve per determinare sequenze temporali o spaziali. Dovendo determinare la sequenza di tre lettere, per

esempio a, b,c, dopo avergli assegnato dei vettori atomici ad a, b, c denominati **a b c** , la sequenza **bac** viene determinata dall'ipervettore formato da: <<(**b**)*<(**a**)***c**.

In questa trattazione opereremo su HV binari ma le stesse considerazioni valgono per HV bipolari. Possiamo quindi definire i processi di classificazione di HDC come 4 fasi distinte (Fig 1):

1) Mappatura: associazione di HV a concetti atomici. Tali vettori sono memorizzati nella memoria degli elementi (IM) e restano immutate per tutto il processo. Vengono impiegati diversi tipi di ipervettori di base, inclusi ipervettori casuali selezionati quasi ortogonalmente dall'iperspazio [12], ipervettori di livello con correlazione lineare e ipervettori circolari con correlazione circolare o ciclica [24].

2) Codifica: Utilizzo delle operazioni matematiche per creare idee, concetti e classi di concetti. La codifica dei dati in HDC comporta la combinazione di due o più ipervettori per creare rappresentazioni di informazioni complesse. Diversi metodi sono impiegati per codificare vari tipi di dati, come il testo dai caratteri [28], i grafici dai vertici e dagli spigoli [29], le serie temporali dai campioni [30] e le immagini dai valori dei pixel [31].
3) Memorizzazione: Memorizzare i diversi concetti, classi, comportamenti. Questi vengono memorizzati nella memoria delle classi o memoria associativa (AM):
4) Inferenza: Codifica di un nuovo comportamento non noto e ricerca nella AM della classe più somigliante.

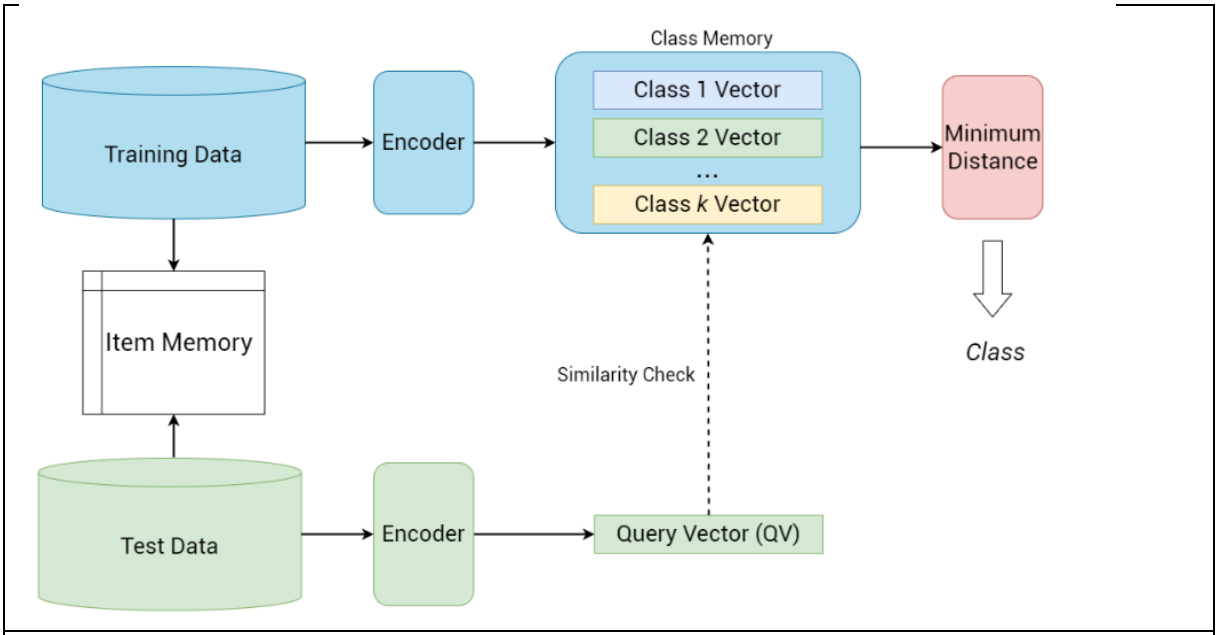


Fig 1: Le fasi di una attività di HDC,

Come si evince dalla figura, tutte le fasi sono ben definite sia matematicamente che temporalmente, La parte più delicata è l'inferenza, ovvero la ricerca nella AM del HV più simile per determinare la classe di appartenenza. Questa operazione richiede diverse operazioni: partire dal primo HV memorizzato, confrontare bit per bit il HV di inferenza con HV di classe, determinare la distanza o somiglianza, passare al secondo HV e fare la stessa operazione fino a comparare l'ultimo HV della AM ed infine scegliere l'HV con la migliore somiglianza.

Descrizione metodo proposto

Quando le classi sono molto numerose questo richiede tempo risorse e in definitiva spreco di energia. Non esiste fino ad ora un algoritmo specifico ed efficiente per una ricerca in un elenco non ordinato, come quello della AM; quindi, in questo documento proponiamo una soluzione tecnicamente valida nel caso in cui la dimensione degli HV siano sufficientemente più grandi (più avanti specificheremo in che senso) del numero di classi, e che permette di bypassare tutti questi passaggi eseguendo la ricerca in una sola operazione con una probabilitá che cresce rapidamente rispetto al numero di dimensioni di avere il risultato coretto, come mostrato in Fig 2 e dalla equazione seguente, vaida sotto l'ipotesi di ipervettori campionati con distribuzione uniforme dal loro dominio

$$P = (1 - 1/2^{(D/n)})^{(n-1)}$$

Dove D é il numero di elementi degli ipervettori e n é il numero di classi. Da un punto di vista interpretativo, la formula ci spiega che la memoria associativa ha due regimi, uno a bassa precisione e uno ad alta precisione in termini di recupero delle informazioni. La transizione tra questi due regimi risulta essere netta, portando al fatto che sopra una certa soglia di dimensioni la probabilitá di corretto recupero dei dati cresce molto rapidamente saturando a 1.

Utilizziamo una proprietà degli HV che si chiama distribuzione densa. Ogni bit dell'HV ha probabilità 0.5 di essere 0 o 1, il che si traduce nella proprietà che mediamente, in un HV ci sono le stesse quantità di 0 e di 1.

Possiamo dividere questo HV in gruppi consecutivi di **n** bit ed osservare che la distribuzione di 0 e 1 si mantiene nei gruppi finché questi siano di dimensioni tali che il numero di bit in ogni gruppo sia significativa (indicativamente, maggiore di 50 elementi per HV di 10000 dimensioni, ovvero il numero di elementi deve essere minore del 5% della dimensione degli HV).

Quindi, se la AM ha **n** classi, dividiamo gli HV di dimensione **D** in **n** gruppi consecutivi a partire dal bit meno significativo di **D/n** bit. Per esempio, se ci sono 25 classi e i HV sono da 10000 dimensioni dividiamo gli HV in 25 gruppi consecutivi a partire dal bit meno significativo da 400 bit ognuno. Qualora la dimensione D non sia un multiplo intero di n, si considererà un arrotondamento all'intero più vicino. Da ora in avanti, per semplicità considereremo il caso D = 10000 e n = 25, ma chiaramente la trattazione può essere generalizzata.

Creiamo ora un HV detto MV (*magic vector*) composto dai primi 400 bit invertiti del primo HV che determineranno i primi bit di MV, poi i secondi 400 bit del secondo HV e così via fino ad arrivare agli ultimi 400 bit dell'ultimo HV. Abbiamo così generato un nuovo HV di dimensione D detto MV che ha delle caratteristiche particolari: se moltiplicato (XOR) con un dei HV della AM genererà un HV che nel gruppo corrispondente alla posizione in memoria del HV scelto, genererà una serie di 1 consecutivi rompendo la simmetria di diffusione densa dei bit bel HV. Contando il numero di 1 in ogni gruppo, quello che ha il maggior numero di 1 è con altissima probabilità (nel senso illustrato dalla formula riportata precedentemente) la posizione in memoria del HV cercato. Quello che abbiamo fatto è spostare il problema dalla misura della distanza alla distribuzione dei bit o dalla ricerca del HV alla sua posizione in memoria.

Un aspetto di notevole rilevanza dell'algoritmo proposto è la sua resilienza al rumore: qualora una frazione dei bit siano corrotti, se la dimensione dell'ipervettore è sufficientemente grande il numero di 1 nel segmento relativo alla classe corretta sarà comunque sufficiente per poterla identificare senza ambiguità. Più avanti analizzeremo questo aspetto sul piano quantitativo.

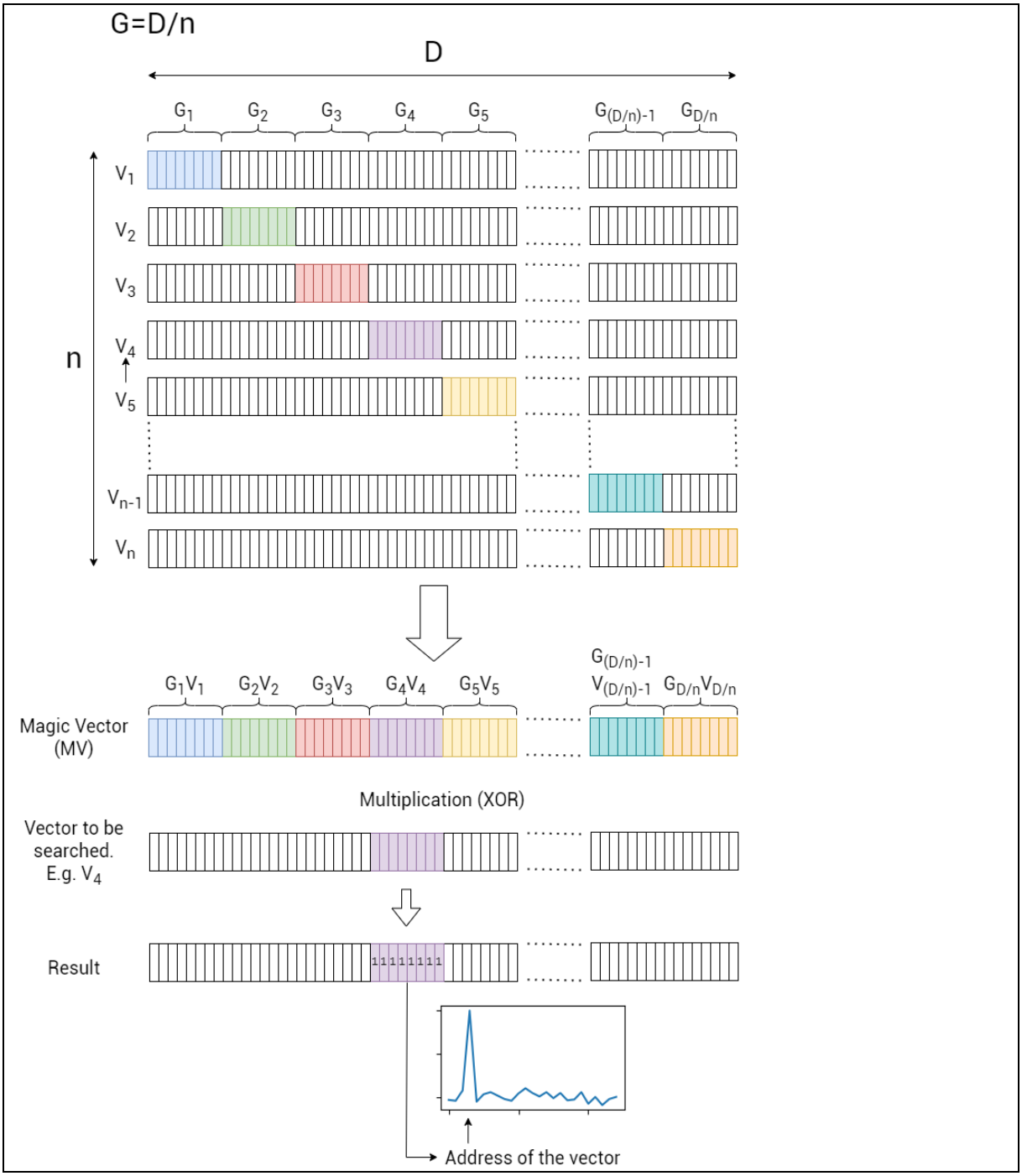


Fig.2. Algoritmo di ricerca ad 1 passo di un ipervettore nella memoria associativa; D=dimensione degli ipervettori, G=Numero di gruppi in cui viene diviso HV: Il prodotto (XOR) del vettore MV per un qualsiasi vettore della AM genera un vettore con tutti 1 nel gruppo ricercato. Il diagramma rappresenta il conteggio di 1 in ogni gruppo.

L'algoritmo sopra proposto è stato validato mediante un software, che ha anche permesso di verificarne l'accuratezza in funzione delle diverse scelte di numero di Item in memoria, numero di dimensioni, degradazione del HV casuale fino al 30% dei bit. I test hanno dimostrato sia la funzionalità che la semplicità di ricerca di un HV su una serie di HV casuali.

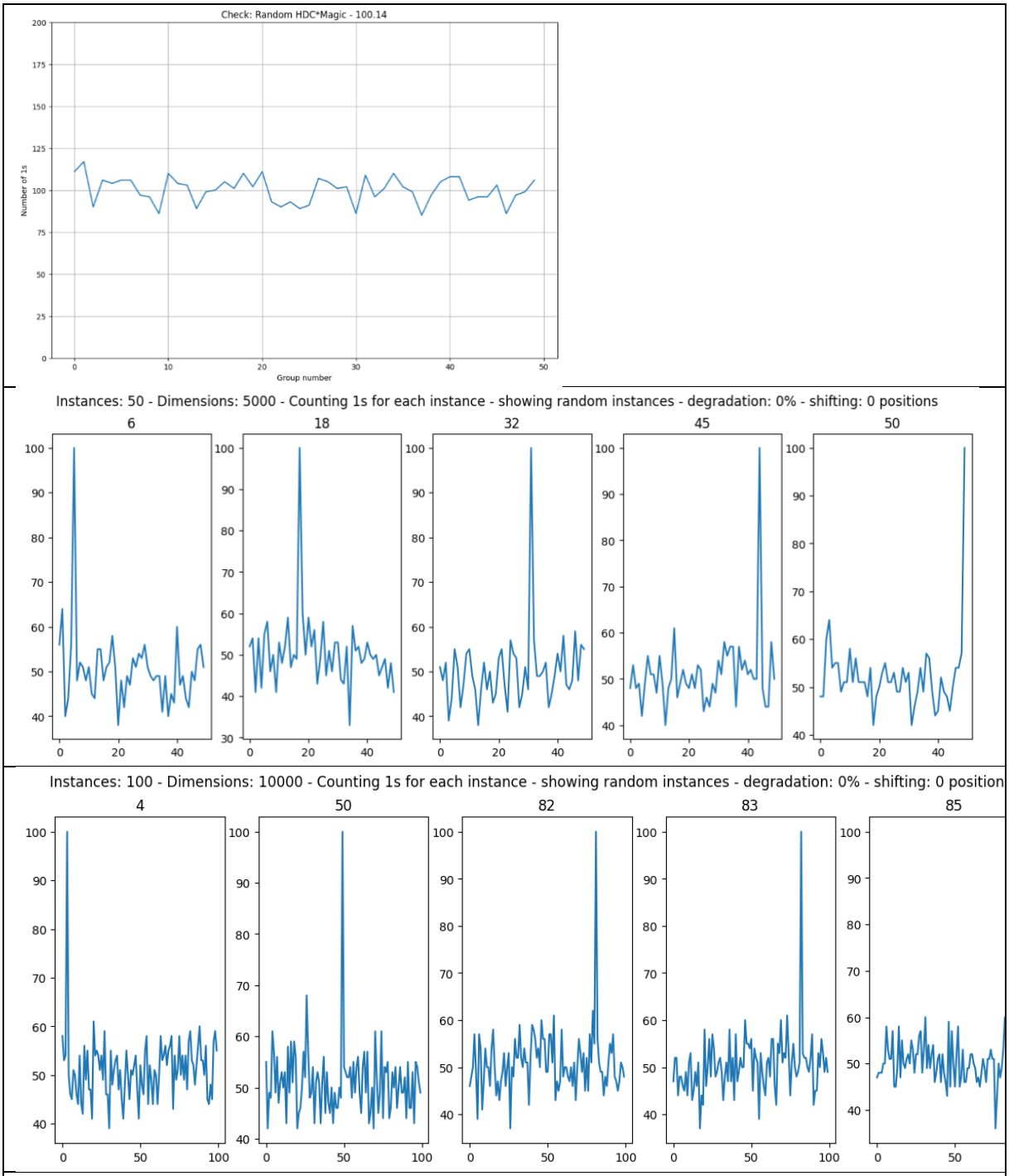


Fig.3. In alto Test su 50 HV in AM con D=5000 e in basso, testo su 100 item in AM con D=10000, Sull'asse Y il numero di 1 e sull'asse X gruppi di bit. Viene preso un HV casuale appartenente alla AM e moltiplicato per MV. Si noti la distribuzione densa degli HV ed il picco in corrispondenza del gruppo di appartenenza del valore cercato

Possiamo alterare gli HV di query per vedere fino a che limite viene riconosciuto per valutare la ricerca di HV somiglianti. I risultati danno risposte accurate fino ad una degradazione del 30% del HV di query, che .

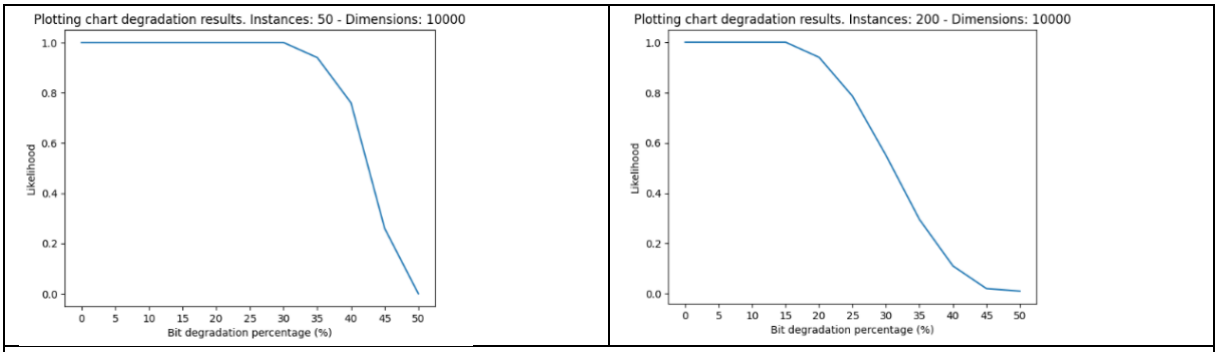


Fig.4 Tasso di errore con ricerca in 50 HV nella AM con dimensione D=10000 a sinistra e 200 HV a destra

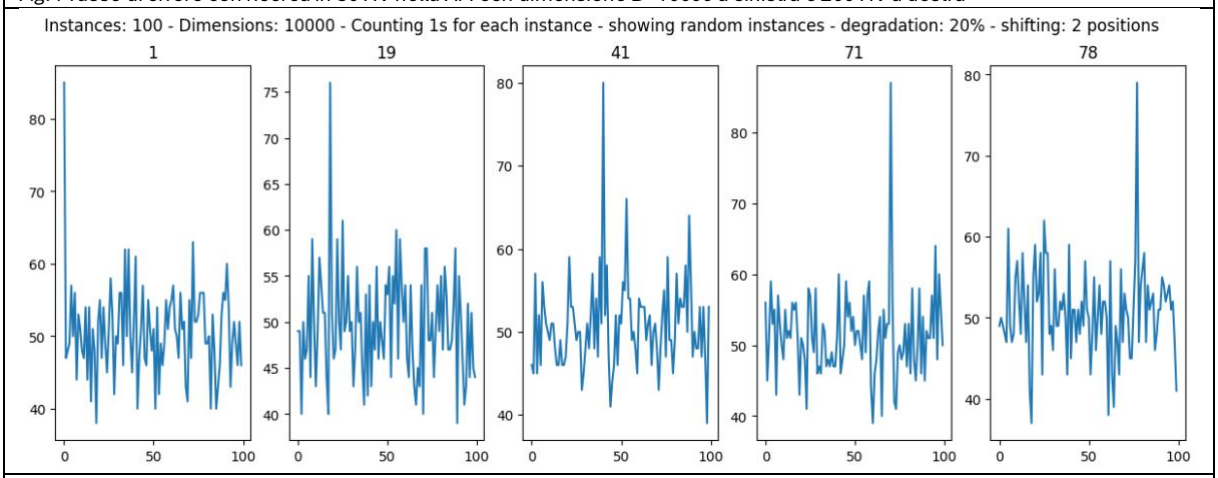
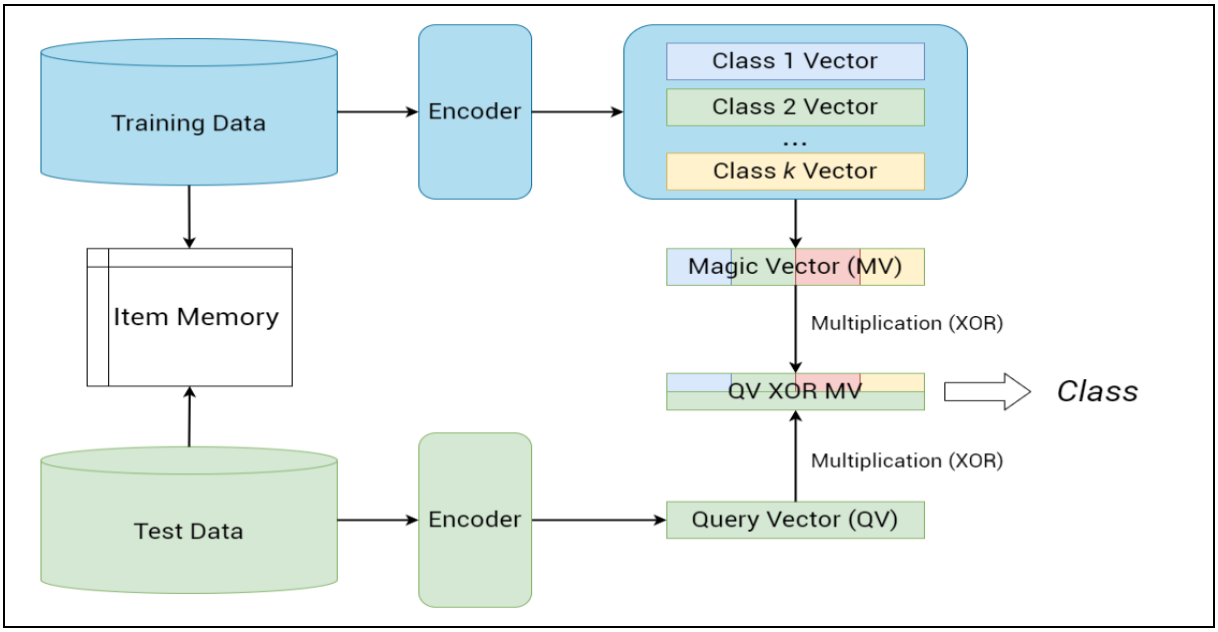


Fig.5 risultati di ricerca di un HV degradato del 20% e con una permutazione di 2 bit a sinistra. Il riconoscimento su 100 HV con D=10000 è ancora ottimo.

Se il numero di elementi nella AM è molto grande, è possibile dividere la AM in due o più gruppi calcolandosi i rispettivi MV ed eseguire la ricerca in più passi oppure in parallelo.

Altra importante osservazione è che, in applicazioni Client-Server, dove l'addestramento è fatto normalmente off-line dal server, non è necessario per il client memorizzare tutti gli HV della AM, ma solo il MV da moltiplicare per il HV di query generato localmente. Il client darà come risposta direttamente la classe di somiglianza, risparmiando memoria sul client.

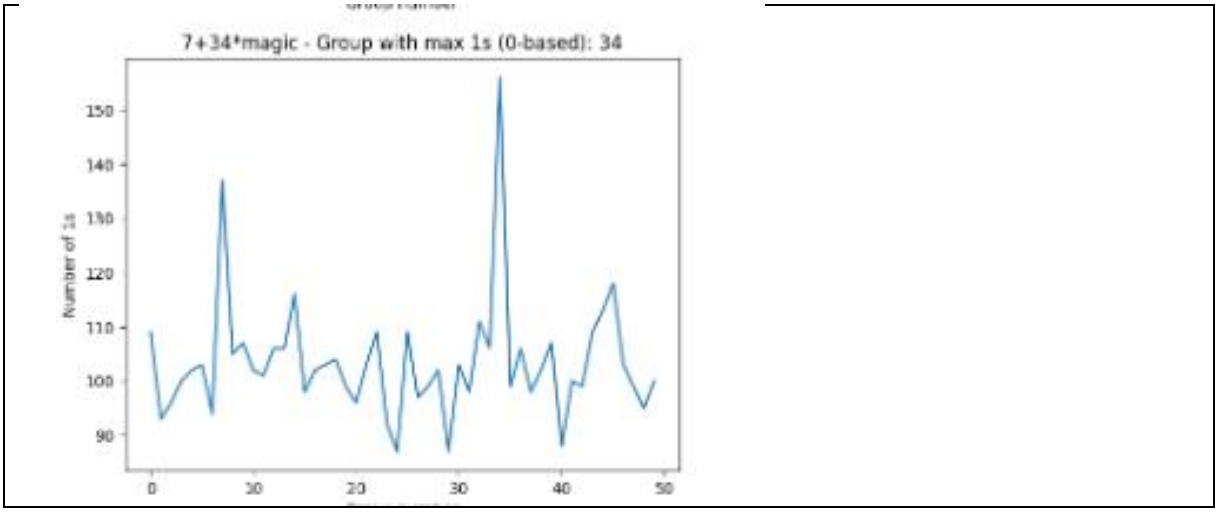


Un'altra applicazione interessante che può essere eseguita con i MV è la ricerca nella memoria degli item atomici, detta anche memoria di pulizia, delle componenti di un HV di cui non si conosce la composizione, Dato un HV, vogliamo verificare se il HV è il risultato di una somma di HV atomici (**S=A+B)**. Ricordando che la somma di due HV genera un HV che è simile (in termini di distribuzione degli "1") alle sue componenti, la procedura standard sarebbe quella di confrontare tale somma con tutti gli HV della memoria e cercare i due più somiglianti. Proponiamo ora un algoritmo alternativo e più efficiente.

Come si vede in Fig.6, invece, moltiplicando il vettore incognito per MV, se le sue componenti fanno parte della memoria degli elementi, questi vengono rivelati. Infatti, in modo simile a quanto considerato sopra, il risultato della moltiplicazione di S per MV presenterà un elevato numero di 1 nei segmenti relativi alle classi A e B. Il conteggio del numero di "1" permette così di identificare gli HV atomici che portano alla somma del HV di partenza.

Per formalizzare matematicamente i passaggi descritti nel testo, possiamo utilizzare la seguente notazione:

formalismo matematico da esplicitare



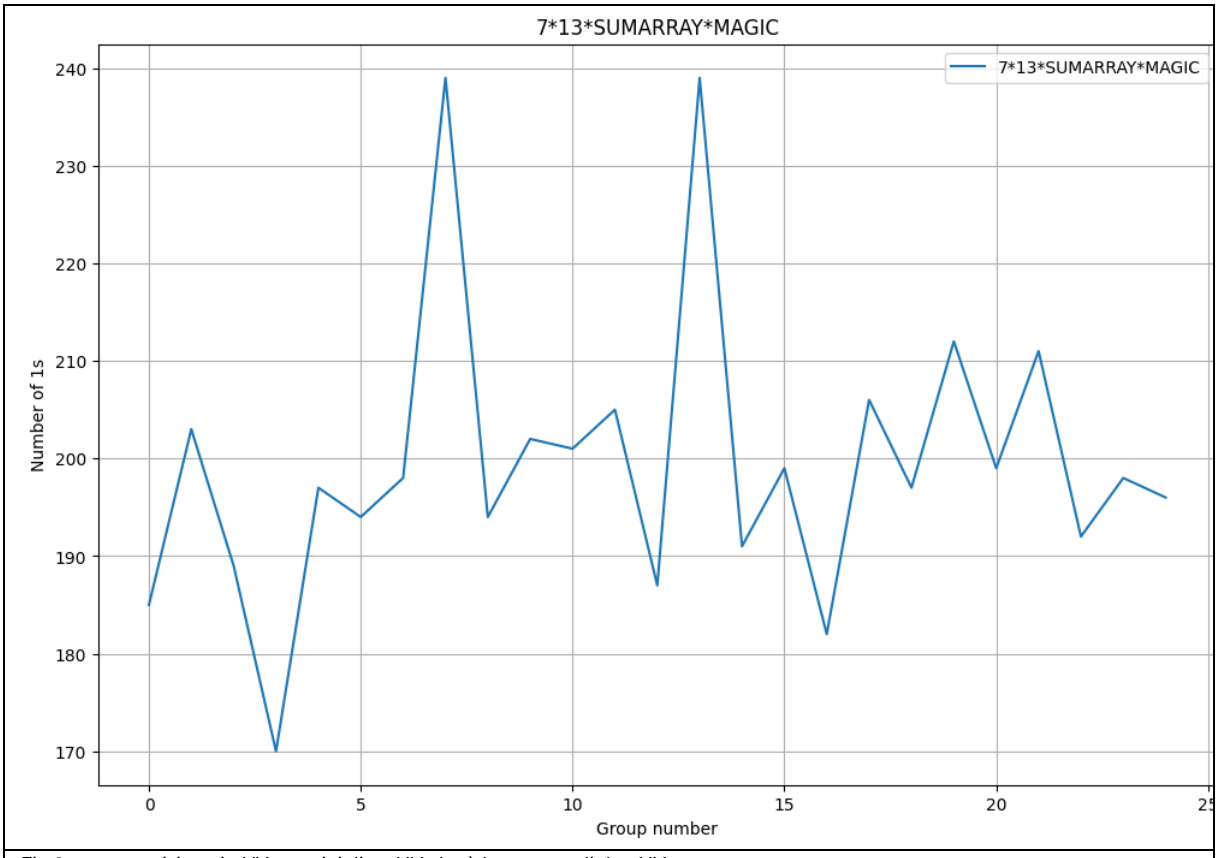


Fig.6: scomposizione in HV atomici di un HV che è la somma di due HV

Questa proprietà è molto importante quando è necessario fare una pulizia di un risultato di una query per ricavare i HV atomici originali.

Una procedura simile, pur con un passaggio ulteriore, è la scomposizione di un prodotto di due o più HV atomici. Detto P = A*B il prodotto da fattorizzare, sappiamo che il prodotto di due HV ne genera uno che è dissimile dai suoi componenti e quindi non è possibile applicare una ricerca di somiglianza. sfruttando la proprietà distributiva della moltiplicazione rispetto alla somma possiamo procedere come segue.

Sia S un HV che rappresenta la somma normalizzata a maggioranza (come descritto precedentemente) di tutti i HV della memoria atomica, allora

S*P = (A+B+C+D+... Z) * A*B

= A*A*B+A*B*B+A*B*C+A*B*D+...+A*B*Z)

= A+B+... (rumore)

Dove nell'ultimo passaggio abbiamo sfruttato l'identità A*A*B = B, e il fatto che il prodotto fra 3 HV scorrelati è assimilabile a una componente di rumore.

Abbiamo quindi riportato il prodotto di due HV alla somma degli stessi più una componente che possiamo definire come rumore e degradazione del vettore risultato. Per ottenere le componenti del prodotto quindi basta eseguire:

MV*S*P

Come mostrato nella Fig. 7, ottenendo immediatamente i componenti del prodotto o, più precisamente, le componenti più probabili, con un livello di confidenza tanto maggiore quanto sarà maggiore il rapporto fra numero di elementi dello spazio HV e del numero di classi.

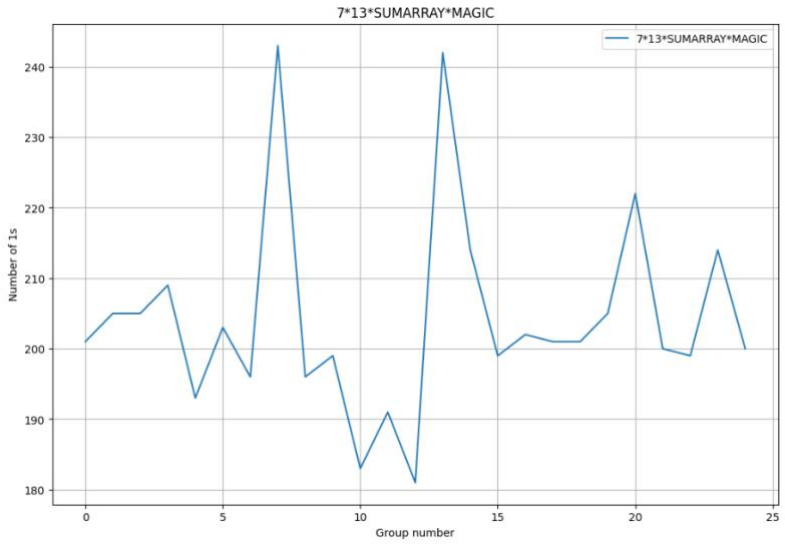


Fig 7. Fattorizzazione di un prodotto di HV e probabilità di trovare il risultato.

Pertanto, può succedere che, in funzione del numero elevato di elementi o di un numero di dimensioni ridotte dello spazio HV, il risultato sia ambiguo e non porti ad una soluzione certa della fattorizzazione come si vede in Fig. 8

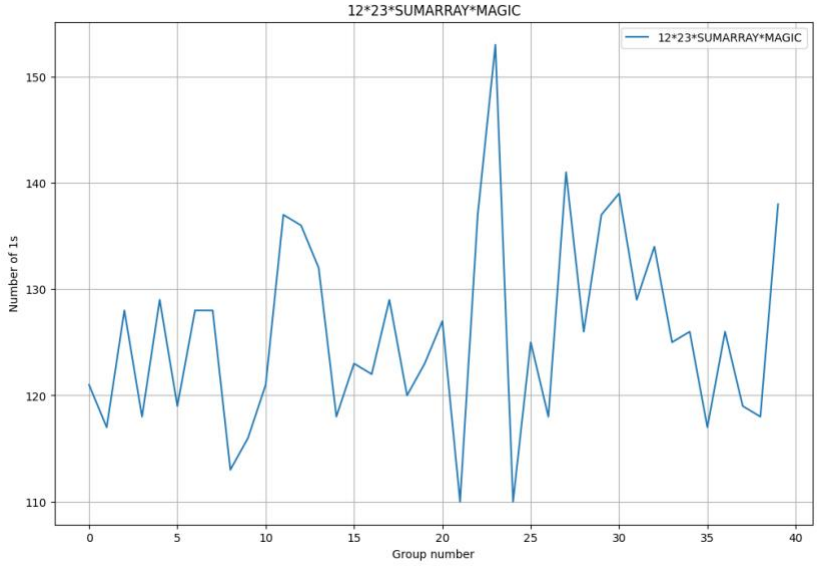


Fig. 8 Risultato ambiguo di una fattorizzazione con AM=40 elementi e D=10000

Si può allora procedere ad una nuova iterazione avendo cura di utilizzare come somma ponderata solo gli HV che superino una certa soglia, che nel nostro caso è la media dei valori sommata alla deviazione standard. In questo modo si diminuisce la componente rumore e la soluzione viene trovata più facilmente, come in fig. 9.

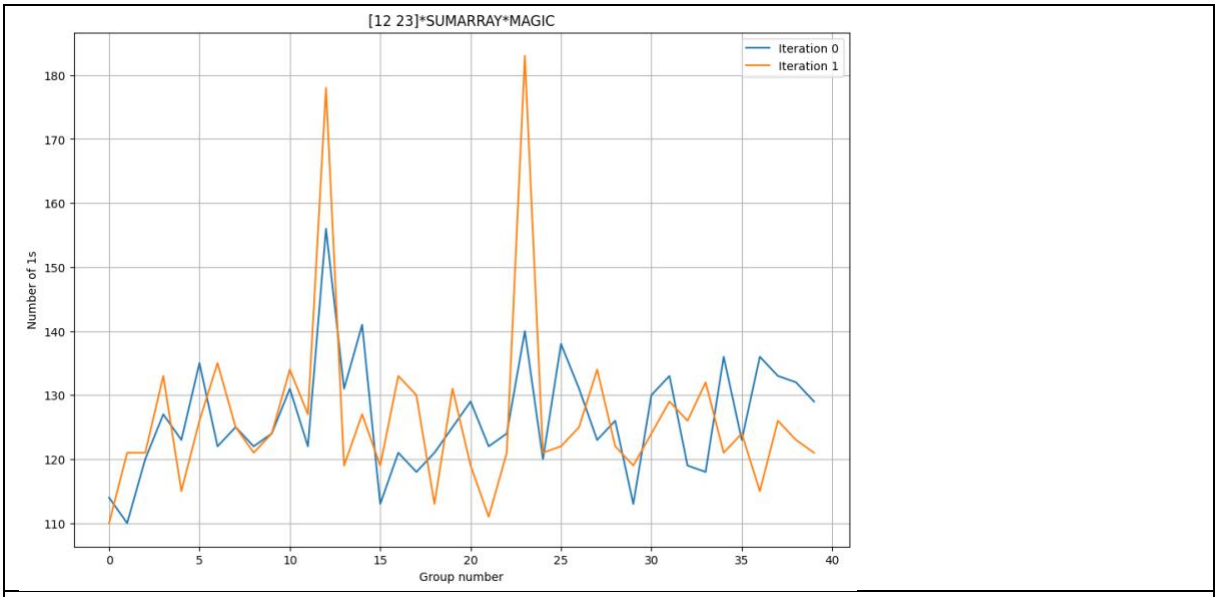


Fig.9 ricerca fattorizzazione prodotto con AM=40 elementi, D=10000, con 2 iterazioni.

Il metodo puo' essere facilmente utilizzato per AM con grandi dimensioni e HV altrettanto grandi: come in Fig.10:

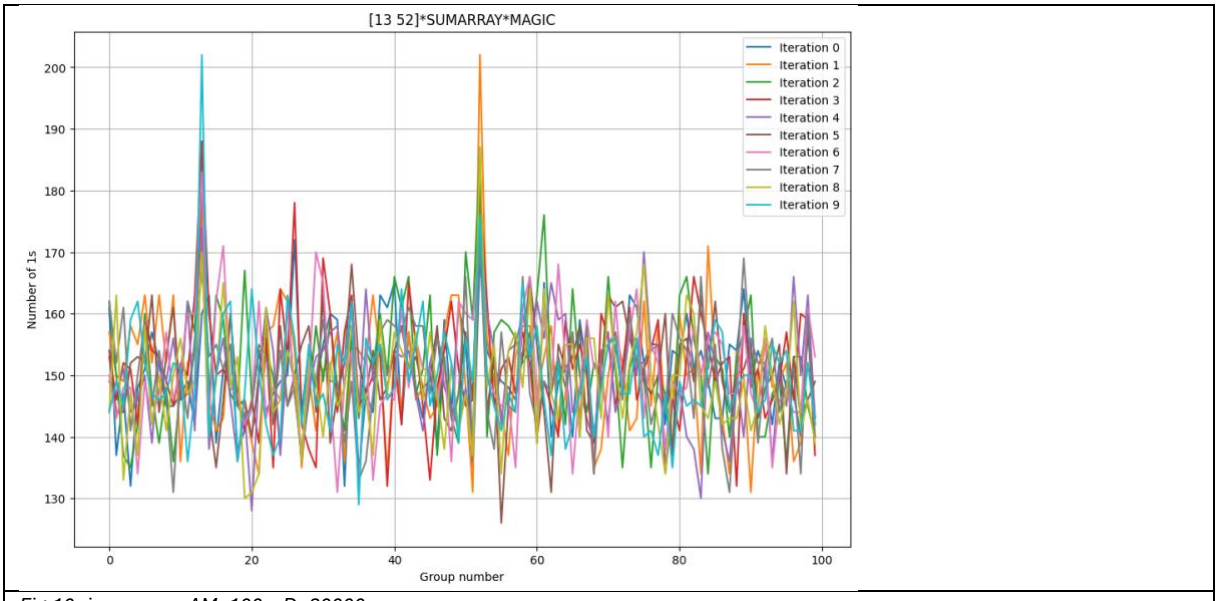


Fig.10 ricerca con AM=100 e D=20000

Per una ricerca della fattorizzazione di piu' elementi derivanti da diversi codebook, in letteratura esistono diversi metodi, uno dei quali e' denominato come Resont Network, Dati per esempio 3 codebbom ognuno di N ipervettori di dimensioni D, si vuole trobare le componenti del vettore s =a(n)*b(m)*c(q) dibe a b c sono tre HV ognuno appartenene al codebook A, B e C.

Detto MVA il Magic vector del codebook, MVB ilquello del codebool B e MVC quello di C, e dettoMide(_a(t) la somma pomderata di tutti i HV del codebbol A, MODEB(t) quello del codebook B e MODEC(t= quello di C si procede come segue:

a(t)= MVA*s*MODE_B(t)*MODE_c(t)

b(t)=NVB*s*MODE_A(t)*MODE_C(t)

c(t)=MVC*s*MODE_A(t)*MODE_B(t).

a(t) b(t) e c(t) sono degli HV speciali che divisi nei gruppi con cui si soono divisi gli MV, contando gli 1 in ogni gruppo, determinano l'indirizzo nel codebool del HV piu' somiglianyr.

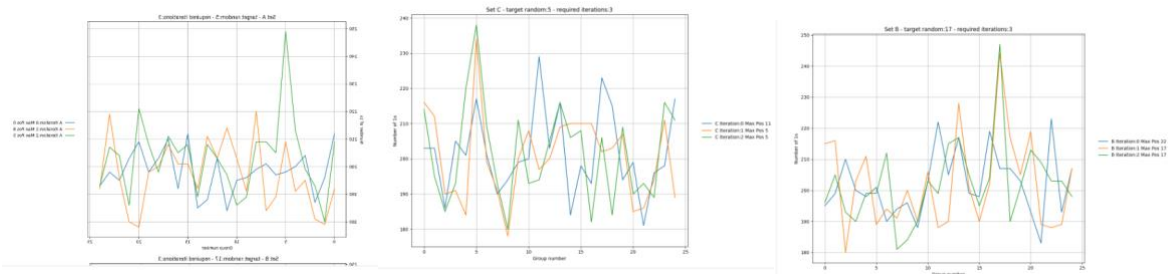


Fig.18 risultati dopo 3 iterazuioni du un prodotto con codebook da 20 HV.

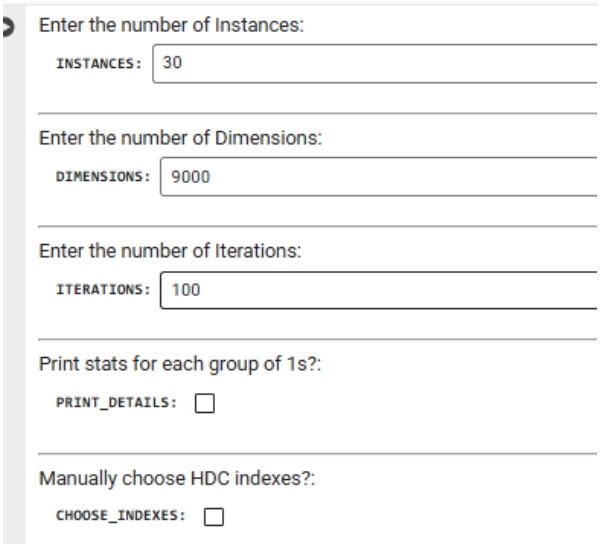
Anche qui, se non si trovasubito una somiglianza ottimale, si procede al secondo step, avendo cura di ricalcolarsi I MODE con gki HV che suoerano una cera soglia data dalla medi +la deviazione standard. , quindi sicalcola MODE:A(t+1) MODE_B(t-1) e MODE_C(t+1(e si riprocede a calcolare:

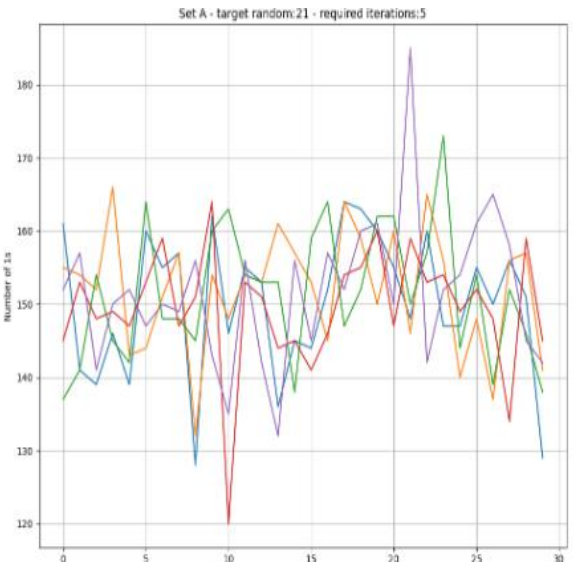
a(t+1)= MVA*s*MODE_B(t+1)*MODE_c(t+1)

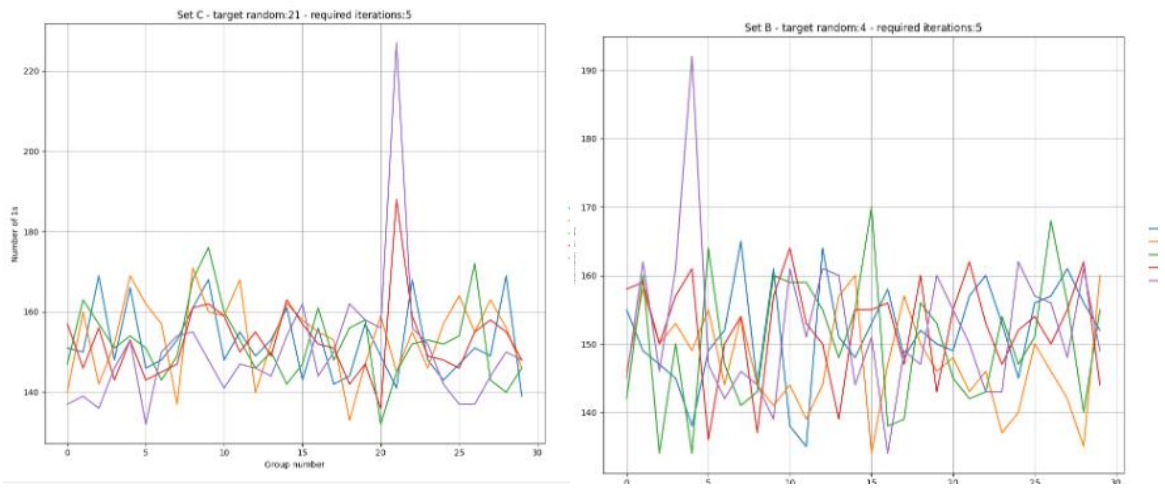
b(t+1)=NVB*s*MODE_A(t+1)*MODE_C(t+1)

c()+1)=MVC*s*MODE_A(t+1)*MODE_B(t+1).

I vantaggi rispetto all'algoritmo classico delle reti di risonatori sono la eliminazione delle ricerche di somiglianza fatte per ogni HV e il miglioramento dell'approssimazione del HVdi sovrapposizione







*Fig… Esempio di ricerca dei fattori di un prodotto s=a(21)*b(4)*c(21) si un sei di 3 codebook da 30 elementi ciascuno*

Menzioniamo ora un'ultima applicazione alla ricerca all'interno di database organizzati per coppie chiave-valore.

Dati un insieme di coppie di *codebook* Key e Value che rappresentano gli item di una serie di elementi che possono essere associati, come per esempio dei nomi (Key) e dei numeri di telefono (Value) è possibile generare un HV che ne rappresenti l'intero insieme, eseguendo le seguenti operazioni:

E= K1*V1+K2*V2+….Kn*Vn

Generando i Magic Vector sia per il codebook Key (MVK) che per il codebook Value (MVV) , è possibile risalire, dato un qualsiasi Key al suo rispettivo Value. Ad esempio, dato un generico codebook key (Ki) associato all'elemento di indice i, abbiamo:

Ki * E = Ki* K1*V1 +Ki*K2*V2+…..Ki*Vn*Kn = Vi + rumore

dove abbiamo applicato lo stesso argomento usato nell'algoritmo di fattorizzazione.

Il risultato è quindi un HV simile a V2, al netto di una certa quantità di rumore. Moltiplicando quindi questo HV per MVV (Magic Vector degli item di valore) si ottiene un HV che, diviso nel numero dei gruppi, darà al gruppo V2 il maggior numero di 1, come si vede dalla figura 11

Experiment 10 - Simulation of research in a key-value database

- Generate a random tensor for the hypervectors K, for the keys
- Generate a random tensor for the hypervectors V, for the values
- Generate the magic vector for the keys K
- Generate the magic vector for the values V
- Obtain the multiplication tensor KxVx
- Obtain the H vector, which is the sum of all the products KxVx –> K1V1 + K2V2 + K3V3.
  The vector represents all the dataset.
- Perform a research of the value associated to the provided key index Kx.
  E.g. K2. Multiply K2 by H
  ==> K1V1 × K2 + K2V2 × K2 + K3V3 × K2 ...
  ==> therefore K2 is eliminated and only V2 remains: ~~K2~~V2 × ~~K2~~ + K1V1 × K2 + ... + K3V3 × K2 ...
  ==> V2 + noise
- Research. Now compare the previous result with values magic vector in order to obtaining the result.

Enter the number of Instances:

INSTANCES: 25

Enter the number of Dimensions:

DIMENSIONS: 10000

Print stats for each group of 1s?:

PRINT_DETAILS: ☑

Plot details for each group of 1s?:

PLOT_DETAILS: ☑

```
Generating keys hypervectors K...
Generated random tensor of 25 * 10000 binary values.
Generating values hypervectors V...
Generated random tensor of 25 * 10000 binary values.
Computing the magic vector for keys...
Computing the magic vector for values...
Computing the KxVx multiplication tensor...
Computing the sum vector H...

Choose the key index (0-based) of the item to be searched.
Finding value for key 5
```

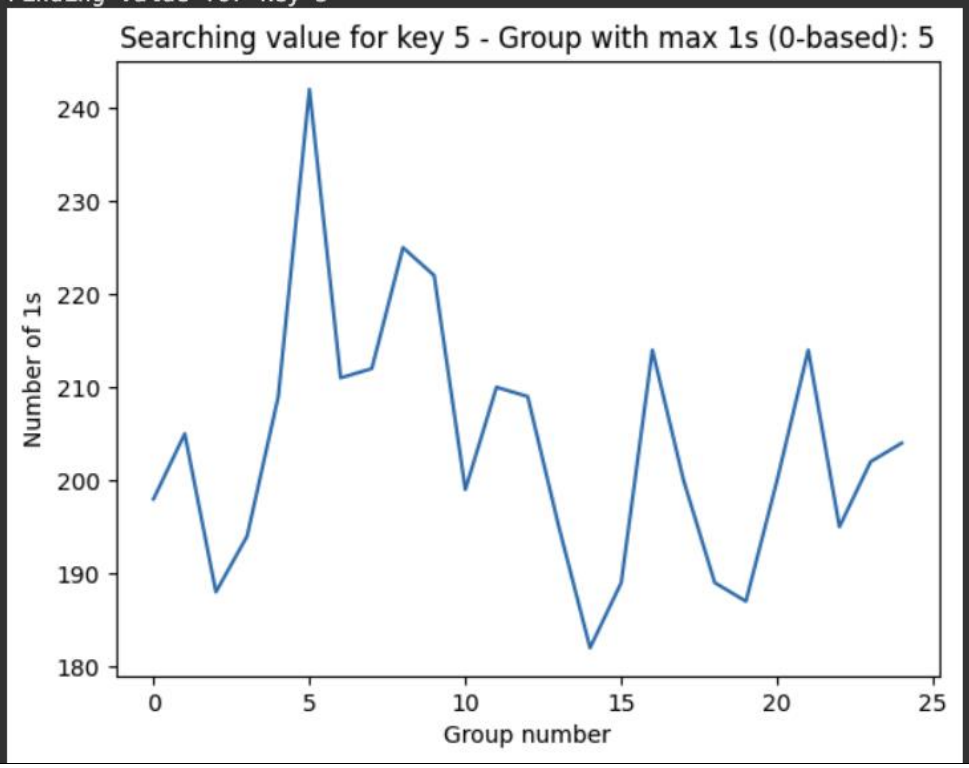


Fig.11 ricerca con AM=100 e D=20000

Implementazione hardware

L'algoritmo oggetto del presente brevetto, relativo alla ricerca di somiglianza e operazioni di fattorizzazione in memoria associativa iperdimensionale, risulta particolarmente idoneo all'implementazione sull'architettura ottica descritta nel brevetto WO 2021/205213 (domanda internazionale PCT/IB2020/053388), intitolato "Parallel optical computing system".

Tale idoneità deriva dalla stretta corrispondenza tra le primitive computazionali richieste dal presente brevetto e le funzionalità proprie del sistema ottico descritto in WO 2021/205213. In detto brevetto è infatti prevista un'architettura composta da moduli ottici programmabili, basata su celle a cristalli liquidi e filtri di polarizzazione, idonea a eseguire in parallelo operazioni elementari su una pluralità di canali ottici. Il presente brevetto, a sua volta, formula il calcolo su ipervettori binari ad alta dimensionalità mediante operazioni semplici e ripetitive, in particolare operazioni bitwise di tipo XOR o binding, operazioni di sovrapposizione con successiva normalizzazione e decisioni basate sulla massima similarità o sull'emersione di picchi statistici.

Sotto tale profilo, l'algoritmo oggetto del presente brevetto si presta in modo naturale a essere tradotto in una elaborazione ottica parallela, poiché ciascuna componente dell'ipervettore può essere associata a un corrispondente canale ottico, mentre la trasformazione complessiva può essere realizzata tramite configurazione programmata degli stati delle celle e delle relative maschere di calcolo. Il vantaggio

computazionale è particolarmente rilevante: l'impiego del cosiddetto Magic Vector (MV) consente infatti di convertire una ricerca che, in una implementazione elettronica convenzionale, richiederebbe una pluralità di confronti seriali, in un singolo passaggio computazionale o comunque in un numero fortemente ridotto di passaggi. In tal modo, il processore ottico sfrutta la propria parallelizzazione nativa per elaborare simultaneamente tutte o gran parte delle componenti del vettore, riducendo latenza, accessi intermedi alla memoria e, in prospettiva, consumo energetico per operazione utile.

Un ulteriore elemento di compatibilità è rappresentato dalla scalabilità. Il presente brevetto trae infatti beneficio dall'aumento della dimensionalità degli ipervettori e dalla possibilità di trattare simultaneamente un elevato numero di elementi memorizzati, mentre il sistema di WO 2021/205213 è stato concepito per operare con matrici di celle, configurazioni modulari espandibili e pluralità di raggi o canali elaborati in parallelo. Ne consegue che l'aumento della numero di unità del problema non comporta necessariamente una crescita proporzionale della latenza, ma può essere assorbito in misura significativa dalla parallelizzazione fisica propria del processore ottico.

Rileva inoltre il fatto che l'algoritmo oggetto del presente brevetto presenta una intrinseca robustezza rispetto a degrado, rumore e alterazioni parziali del vettore di query. Tale caratteristica assume particolare importanza ai fini di una implementazione hardware reale, e in particolare ottica, nella quale inevitabili non idealità di modulazione, sensibilità del rivelatore e tolleranze di allineamento possono introdurre perturbazioni sul segnale. La natura distribuita della rappresentazione iperdimensionale, unita a criteri decisionali fondati su regole di maggioranza o su massimi statisticamente significativi, consente di mantenere affidabilità operativa anche in presenza di errori locali o degradazioni moderate.

La medesima considerazione vale anche per le ulteriori funzionalità descritte nel presente brevetto, quali la fattorizzazione di composizioni, la ricerca in codebook multipli e il recupero di associazioni in strutture key-value. Anche tali elaborazioni riutilizzano le stesse primitive di base e possono pertanto essere implementate sulla piattaforma di WO 2021/205213 mediante riconfigurazione dei pattern di calcolo e dei ruoli funzionali dei moduli. Deve peraltro osservarsi che l'algoritmo del presente brevetto non è limitato all'hardware OPTOPC, ma può essere eseguito, con i dovuti adattamenti implementativi, anche su altre piattaforme atte a supportare operazioni binarie e/o ad alta parallelizzazione, quali architetture elettroniche convenzionali, FPGA, ASIC, memorie computazionali in-memory, dispositivi fotonici integrati o sistemi ibridi optoelettronici. L'architettura di WO 2021/205213 costituisce tuttavia una realizzazione particolarmente vantaggiosa, poiché allinea in modo diretto struttura fisica del calcolo e struttura matematica dell'algoritmo.

Note di Giovanni: Applicazioni

- Ottimizzazione processo produttivo in un impianto industriale composto da:
    - N unità produttiva in grado singolarmente di realizzare il prodotto finito K.
    - Dove il prodotto finito ha M varianti:
        - D-Dimensioni: diametro di base, conicità, altezza,
        - C-Colore materia prima: tinta unità, più colori
        - A- Accessori da realizzare in macchina:
            - AS-stampe: piano, rilievo, colori differenti
            - AD-decorazioni: perforazioni, apertura
    - Ogni macchina cambia lavorazione, ossia prodotto finale, da 0 a 3 volte per giorno (h24)
    - Il rendimento del processo aumenta se le macchine devono eseguire con continuita prodotti simili, ossia se la macchina N2, dopo aver realizzato il lotto K4 =f(Dn,Cn,ASn,ADn)

Idee di applicazioni:

- Classificazione di dati alto-dimensionali. Si tratta di un problema tecnologico che trova applicazioni in svariati ambiti, fra cui
    - Classificazione di dati medici (come pattern genomici o segnali elettrocardiografici). La ricerca rapida di somiglianze permette di accelerare l'analisi di dataset biologici, migliorando potenzialmente diagnosi e personalizzazione delle terapie.
    - Analisi di feed video o dati sensoriali per identificare anomalie o comportamenti specifici. Esempi: riconoscimento di movimenti sospetti in ambienti ad alta sicurezza, classificazione di eventi come incendi o intrusioni attraverso reti di sensori. L'algoritmo potrebbe consentire di classificare grandi flussi di dati in real-time, fondamentale per sistemi critici.

La domanda è: è troppo generico come ambito? Va circostanziato a un contesto più preciso?

Inoltre nei casi sopra andrebbe stabilito come definire l'encoding.

Un caso applicativo che potrebbe risultare interessante a nostro avviso risiede nella cosiddetta similarity search nel contesto degli embedding prodotti dalle reti neurali. Nel machine learning moderno, un paradigma assai diffuso di codifica dell'informazione si basa sull'idea di produrre rappresentazioni a dimensione elevata di dati (immagini,testo, video...). Tale rappresentazione segue le medesime proprietá di quasi ortogonalitá citate nel caso di ipervettori distribuiti uniformemente [Lidenstrauss-Johnson Lemma, Vershynin http://math.uci.edu/~rvershyn/papers/HDP-book/HDP-book.pdf]. Alla luce del fatto che molte tecnologie moderne che fanno uso di sistemi di AI hanno come motore principale la ricerca per similaritá tra questi embedding (come ad esempio i sistemi di Retrieval Augmented Generation) reputiamo interessante da un punto di vista computazionale la possibilitá di sfruttare l'algoritmo proposto per accelerare in maniera significativa tali operazioni.