**TRACCIA**

Algoritmo intelligente per la verifica di numeri primi gemelli.

In questo algoritmo utilizziamo la formulazione di primi gemelli, cioè con p e q numeri primi, in una modalità più agevole da un punto di vista computazionale.

Nello specifico abbiamo visto che fissato k due numeri *p*1 e *p*2 sono gemelli se sono del tipo

e con *k* tale che:

∞

*i*=1

*(Formula* ***a)***

La struttura degli *ri* ha mostrato che ad eccezione di 1 e 2 che sono sempre inclusi, i *k* devono essere sempre multipli di 3 per soddisfare *r*1 ed mentre per soddisfare *r*2 devono essere numeri il cui l’ultima cifra termina per {0, 1, 4, 5, 6, 9}.

La condizione **(a)**, quindi,può essere ridotta in modo da avere un algoritmo efficace.

Infatti, piuttosto che avere l’intersezione infinita di *ri* sarà sufficiente fermarsi ad *rj* con ***j***dato come segue al fine di verificare se il k assegnato, chiamandolo K (*con tilde sopra, da ora userò il simbolo* ***H*** *per indicare questo simbolo)* sarà contenuto nei primi *rj* sarà certamente contenuto nei successivi *rj+1* | *rj+2* | *… | rj+n* dati.

La forma degli *ri* in **(12)**.

La determinazione di *j* si ottiene come:

*(Formula* ***b****)*

Che segue banalmente da **(12)**.

Ciò, quindi, ci permette di limitare il numero di *ri* da utilizzare nell’intersezione.

Inoltre, la stessa lunghezza, dipendente da ***m***, a ogni singola *ri* può essere limitata in prima approssimazione come:

Ovvero, osservando che H per essere utile deve certamente appartenere a ,

utilizzando la condizione:

Grazie a queste considerazioni la ***(a)*** diventa:

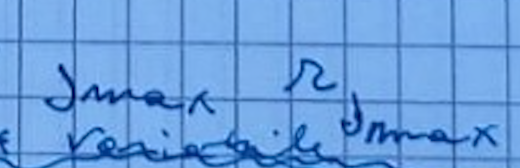
*i*=1

Dove

Tale algoritmo di ricerca dei H associati a numeri primi gemelli può essere ancora ulteriormente ottimizzato scegliendo un .

Nello specifico abbiamo:

*(Formula )*

In tal modo, invece di considerare elementi, se ne otterrà un set minore, ottimizzato di set, ognuno di lunghezza variabile

Vediamo l’algoritmo per verificare se un H genera *twin step per step*:

1. Accetta input H
2. Calcola *formula* ***b***
3. Genera secondo **(12)**
4. Fissato calcola secondo *()*
5. Se

* Incrementa i
* Ripeti il passaggio 4.
* Se
  + Calcola fin quando esso non è maggiore o uguale ad H.
  + Se = H allora incrementa
  + Se allora H non genera numeri primi gemelli
* Se
  + Calcola fin quando esso non è minore o uguale ad H.
  + Se Calcola allora incrementa
  + Se Calcola allora allora non genera numeri primi gemelli

1. Se calcola, salva e mostra

Dove #(K) è il numero di K valutati.

Il vantaggio di tale algoritmo di machine learning, rispetto ad algoritmi tradizionali, è che un algoritmo tradizionale genererebbe una matrice con almeno elementi.

Viceversa, il presente algoritmo, in un’ottica di matrici sparse, calcola solo alcuni elementi di .

In particolare, per ogni riga, cioè al variare di , si calcola solo l’elemento K nella posizione di colonna ed eventualmente i suoi primi vicini.

La figura seguente fornisce uno schema concettuale di quanto descritto:

Immagine che contiene linea, testo

Descrizione generata automaticamente

Gli esperimenti condotti hanno riguardato .

Qui di seguito riportiamo la tabella di comparazione tra un algoritmo tradizionale e quello qui presentato al variare di H:

|  |  |  |
| --- | --- | --- |
| H | Algoritmo classico | Algoritmo intelligente |
| 1 |  |  |
| 10 |  |  |
|  |  |  |
|  |  |  |
| … |  |  |
|  |  |  |

N.B. vale per tutte le potenze di 10.

Il seguente grafico mostra al variare di H la curva dei K calcolati sia nel caso di un algoritmo tradizionale, sia nel caso del presente algoritmo intelligente.

La figura successiva, inoltre, mostra al variare di H sia il grafico della frazione utile di K calcolati riaspetto ai , cioè:

Sia il grafico della frazione risparmiata, cioè: