ALMA MATER STUDIORUM – UNIVERSITÁ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

Sviluppo di algoritmi di Grid Data Clustering basati su metodi statistici

Tesi in

TECNOLOGIE E SISTEMI PER IL DATA MINING LM

Relatore
Moro Gianluca

Presentata da
Manduchi Gabriele

Sessione III
Anno Accademico 2012/2013

Indice generale

Introduzione....5
Stato dell'arte....7
Data Mining....7
Il task del clustering....9
Lo stato attuale del task del clustering [25]....10
Esempio di un classico algoritmo di clustering: DBSCAN....11
Subspace Clustering -- spazi ad alta dimensionalità....13
Algoritmi grid-based....15
CLIQUE[4]....15
MAFIA[11]....16
Algoritmi density-based....19
SUBCLU....19
INSCY....20
DOC[10]....21
MINECLUS [11]....22
Algoritmi Top-Down....23
Proclus [7]....23
FINDIT....24
Algoritmi per grandi moli di dati....28
BIRCH....28
Grin Algorithm....29
GENIC....32
IHC....33
Mafia vs FindIt....34
CONCLUSIONI....37
Confronto tra algoritmi ibridi , e density-based....37
CONCLUSIONI....39
Statistica applicata al clustering – Hopking 40
I limiti del task di clustering[25]....41
Richiami teorici 45
Casualità dei punti e Distribuzione Binomiale....45
Distribuzione Ipergeometrica e distribuzione ipergeometrica Multivariata....49
Test di verifica di ipotesi applicati alle distribuzioni.[27]....50
Test Chi-Quadro [28]....52
Test di Kolmogorov-Smirnov [29]....54
Algoritmi di clustering basati su metodi statistici....57
Concetti di regione casuale, Densa+ , Densa -....58
Orientamento Grid-Based ed adiacenza d'intorno....58

Suddivisione dei punti all'interno di una regione....59
Adiacenza d'intorno....62
Algoritmo 1 63
Preparazione del campione casuale....64
Preparazione del campione osservato....65
Test di conformità tra distribuzioni....65
Ricorsione....66
Problematiche 66
Algoritmo 2 : Densità e distribuzione....67
Preparazione del campione casuale....69
Preparazione del campione atteso....73
Individuazione regioni Dense +/-....74
Algoritmo 3: Calcolo della probabilità con dipendenza 75
introduzione....75
introduzione....75
introduzione....75
Preparazione del campione casuale....76
Preparazione dello stack di regioni....77
Algoritmo di assegnazione probabilità dipendente ricorsivo....77
Gestione dei buchi....79
Individuazione regioni NON CASUALI +/-....79
Clustering 80
Test , confronti e valutazioni....82
Algoritmi di Confronto 82
Indice di Rand [35]....83
Test DatasetS_ConRumore....85
Test Galaxy....102
Test GirandBisBis....110
Test ex2m50....117
Test T7.10K....125
Test T4.8K....129
Test Compound....133
Test Aggregation....136
Test Flames....139
Test D31....142
Test Line&Cluster....145
Test Dataset4....149
Considerazioni finali....152
Conclusioni....154

Sviluppi futuri.....154
Bibliografia.....156

Introduzione

Organizzare I dati in insiemi significativi è una delle più importanti modalità di estrazione della conoscenza.

Ad esempio uno schema comune di classificazione può essere quello utilizzato dalle scienze naturalistiche , ove tutto il regno animale dei vertebrati è suddiviso in “classi” quali umani,uccelli,rettili,anfibi e pesci
L’analisi di cluster è uno studio formale di metodi ed algoritmi per raggruppare oggetti in base alle caratteristiche intrinseche dei dati che sono simili tra loro.
L’analisi di cluster non utilizza etichette di categoria, e questa mancanza di informazioni a priori differenzia tale ramo da altri meccanismi di estrazione di conoscenza quali alberi di classificazione o analisi di discriminazione (metodologie di estrazione della conoscenza “supervisionate”) .
Lo scopo del clustering è quindi quello di individuare strutture nei dati significative.
Ed è proprio dalla seguente definizione che è iniziata questa attività di tesi , fornendo un approccio innovativo ed inesplorato al cluster , ovvero non ricercando la relazione ma ragionando su cosa non lo sia.
Osservando un insieme di dati ,cosa rappresenta la non relazione ?
Una domanda difficile da porsi , che ha intrinsecamente la sua risposta, ovvero l’indipendenza di ogni singolo dato da tutti gli altri.
La ricerca quindi dell’indipendenza tra i dati ha portato il nostro pensiero all’approccio statistico ai dati , in quanto essa è ben descritta e dimostrata in statistica.
Ogni punto in un dataset, per essere considerato “privo di collegamenti/relazioni” , significa che la stessa probabilità di essere presente in ogni elemento spaziale dell’intero dataset.
Matematicamente parlando , ogni punto P in uno spazio S ha la stessa probabilità di cadere in una regione R ; il che vuol dire che tale punto può CASUALMENTE essere all’interno di una qualsiasi regione del dataset.

Da questa assunzione inizia il lavoro di tesi, diviso in più parti.

- Il secondo capitolo analizza lo stato dell'arte del clustering , raffrontato alla crescente problematica della mole di dati , che con l'avvento della diffusione della rete ha visto incrementare esponenzialmente la grandezza delle basi di conoscenza sia in termini di attributi (dimensioni) che in termini di quantità di dati (Big Data).
- Il terzo capitolo richiama i concetti teorico-statistici utilizzati dagli algoritimi statistici implementati.
- Nel quarto capitolo vi sono i dettagli relativi all'implementazione degli algoritmi , ove sono descritte le varie fasi di investigazione ,le motivazioni sulle scelte architetturali e le considerazioni che hanno portato all'esclusione di una delle 3 versioni implementate.
- Nel quinto capitolo gli algoritmi 2 e 3 sono confrontati con alcuni algoritmi presenti in letteratura, per dimostrare le potenzialità e le problematiche dell'algoritmo sviluppato , tali test sono a livello qualitativo , in quanto l'obbiettivo del lavoro di tesi è dimostrare come un approccio statistico può rivelarsi un'arma vincente e non quello di fornire un nuovo algoritmo utilizzabile nelle varie problematiche di clustering
- Nel sesto capitolo saranno tratte le conclusioni sul lavoro svolto e saranno elencati i possibili interventi futuri dai quali la ricerca appena iniziata del clustering statistico potrebbe crescere.

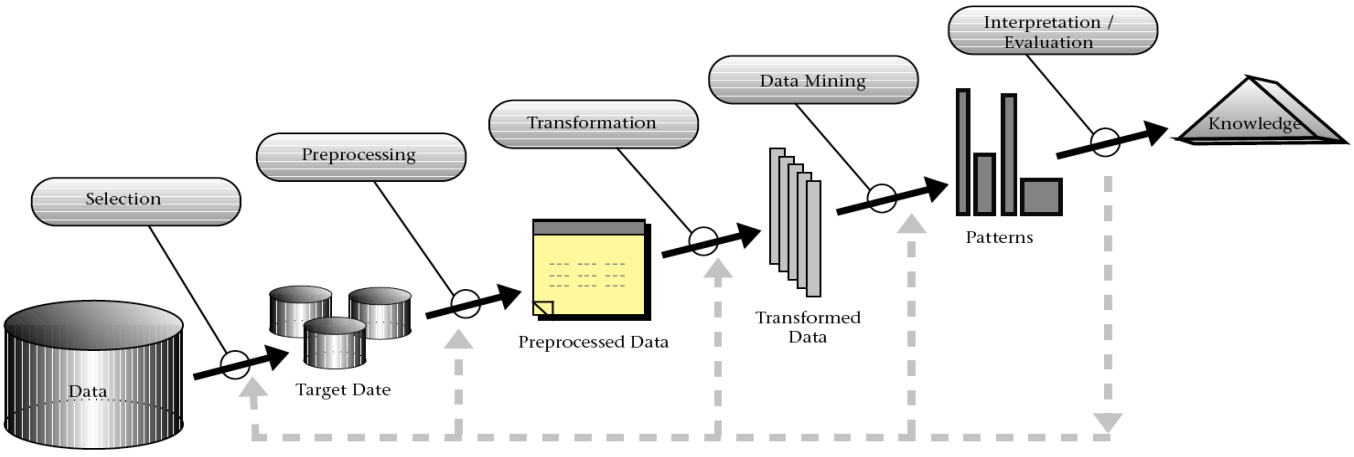
Stato dell'arte

In questo capitolo verrà introdotto il concetto “clustering” dal punto di vista teorico. Dapprima verrà data una definizione del contesto in cui il clustering va a inserirsi, il Data Mining, per poi passare alla definizione e alla spiegazione delle diverse tecniche di clustering dando particolare enfasi a quelle alla base dell'algoritmo sviluppato per questo lavoro di tesi.

Data Mining

Il Data Mining [18] e un insieme di tecniche e metodologie che hanno lo scopo di estrarre informazioni utili da grandi quantita di dati. Il Data Mining costituisce la parte di modellazione del processo di Knowledge Discovery in Database, ovvero l'intero processo di estrazione di informazione dai dati, che va dalla selezione e pre-processamento dei dati fino all'interpretazione e valutazione del modello ottenuto dal processo di Data Mining.

2.1.2. KDD [3]



In figura viene mostrato il processo di Knowledge Discovery; questo si suddivide in 6 fasi, le quali hanno una serie di dati in input e restituiscono in output i dati processati:

- **Selezione**: in questa fase viene creato il dataset obiettivo attraverso la selezione dai dati d'origine di un sottoinsieme di dati su cui eseguire il processo di discovery
- **Pre-processamento**: in questa fase vengono eseguite diverse operazioni atte alla “pulizia" del dataset obiettivo come ,ad esempio, la rimozione di eventuali dati non rilevanti (o rumore) o la decisione della strategia da utilizzare per gestire eventuali dati sparsi.
- **Trasformazione**: in questa fase vengono selezionati gli attributi rilevanti per rappresentare il dataset; questi dipendono dall'obiettivo che si vuole raggiungere e sono usate tecniche di riduzione della dimensionalità.
- **Data-mining**: questa e la fase in cui sul dataset (pre-processato nelle fasi precedenti) vengono ricercati pattern significativi.
- **Interpretazione/Valutazione**: in questa fase si cerca di interpretare i pattern ottenuti dalla fase di data-mining per dare loro un significato.

Parlando piu nel dettaglio della fase 4 del KDD, esistono diverse tecniche di data-mining, queste si suddividono in:

- **supervisionate:** l'algoritmo deve apprendere il comportamento di alcune variabili target .
- **non supervisionate:** le variabili target non sono conosciute a priori.

Tra le tecniche supervisionate troviamo:

- Alberi decisionali
- Reti bayesiane
- Reti neurali

Tra le tecniche non supervisionate troviamo invece il clustering e le regole associative, parleremo ora piu in dettaglio della prima.

Il task del clustering

Il clustering [19] è una tecnica di analisi che ha lo scopo di individuare gruppi significativi in un dataset. La parola “significativi" in questo contesto ha un'accezione puramente topologica: un cluster viene denito come un insieme di oggetti tale che la similitudine tra gli oggetti all'interno del cluster sia maggiore della similitudine tra oggetti interni e quelli esterni al cluster" [19].
Gli algoritmi di clustering possono essere suddivisi secondo le relazioni tra i cluster trovati:

- **partitivi**: trovano cluster disgiunti, ovvero nei quali ogni oggetto appartiene esclusivamente ad un unico cluster. Questi algoritmi possono essere a loro volta suddivisi a seconda del criterio di clustering:
  - **rilocativi**: cercano di trovare i cluster riposizionando gli oggetti iterativamente
  - **probabilistici**: utilizzano distribuzioni probabilistiche per modellare i cluster (es. distribuzioni normali multivariate per l'algoritmo EM)
  - **basati su centroidi**: utilizzano degli oggetti rappresentativi per ogni cluster (chiamati centroidi)
  - **basati su densità**: cercano di individuare regioni dense dello spazio, questi algoritmi sono particolarmente indicati per i cluster di forma convessa

- **gerarchici**: trovano insieme di cluster annidati organizzati ad albero (dendogramma), ogni oggetto in questo caso puo appartenere contemporaneamente a piu cluster nella gerarchia. Questi possono essere a loro volta suddivisi a seconda della strategia di costruzione:
  - **agglomerativi**: utilizzano una strategia bottom-up, inizialmente ogni oggetto e un cluster e si procede via via unendo i cluster

    - **divisivi**: utilizzano una strategia top-down, inizialmente l'intero dataset forma un unico cluster e si procede via via dividendo in cluster diversi.
- **Grid-based**: non utilizzano direttamente i dati ma effettuano una quantizzazione dello spazio in regioni e lavorano sulle regioni invece che sui dati.

- **metodi basati sulla co-occorrenza di dati categoriali**
- **clustering di dataset ad alta dimensionalita o subspace clustering**: algoritmi nei quali i cluster trovati possono essere sovrapposti ma risiedono in sottospazi diversi.

Lo stato attuale del task del clustering [25]

L'esplosione delle informazioni delle comunicazioni anni 2000 e dall'avvento del web Social non solo ha creato un quantitativo enorme di dati, ma li ha anche diversificati sia quelli strutturati che non.
I dati non strutturati sono una collezione di oggetti che non seguono una specifica formattazione (immagini, testo,audio,video, etc) .
D'altra parte , nei dati strutturati vi è da tenere conto la relazione semantica tra gli attributi dello stesso.
La maggior parte degli approcci al clustering utilizzano un vettore la cui rappresentazione è la stessa sia per i dati strutturati che non.
La vista tradizionale però non sempre rappresenta un buon "framework" sulla quale lavorare.
I nuovi modelli e gli algoritmi devono essere sviluppati per affrontare una enorme quantità di dati etereogenei.

Esempio di un classico algoritmo di clustering: DBSCAN

DBSCAN [20] e un algoritmo di clustering basato su densità, in particolare fa parte dell'insieme degli algoritmi basati su connettivita ovvero che computano la densita sui singoli oggetti del dataset. L'altra categoria di algoritmi basati su densita sono quelli density-function che invece cercano di calcolare una funzione di densita sull'intero dataset.

Di seguito alcune definizioni usate da DBSCAN:

- **neighborhood di un oggetto** p: Sia d(p; q) la distanza euclidea tra due oggetti p e q, definiamo con N(p) il numero di oggetti posizionati entro un raggio E da p

$$N_\epsilon(p) = \{q \in D | d(p,q) \leq \epsilon\}$$

- **direct density-reachability**: Sia MinPts un parametro in input che indica il numero minimo di oggetti che un cluster deve contenere. Un oggetto p è directly density-reachable da un punto q rispetto ad a se q è ad una distanza inferiore di E da p ed esistono abbastanza punti nel vicinato contornato da E di p.

$$q \in N_\epsilon(p)$$
$$|N_\epsilon(p)| \geq \text{MinPts}$$

- **density-reachability:** un oggetto p è density-reachable da un punto q rispetto a E e MinPts se esiste una catena di punti direct density-reachable tra i due punti tale che colleghino p a q:

  $$\exists p_1, \cdots, p_n, p_1 = q, p_n = p \text{ t.c. } p_{i+1}$$

  Se tale condizione è soddisfatta allora è directly density reachable da pi .
  Questa relazione non è simmetrica, infatti potremmo avere che q non abbia abbastanza punti da essere considerato denso e quindi varrebbe la relazione da p a q ma non da q a p, si va quindi a introdurre il concetto di density-connectivity.

- density-connectivity: un oggetto p è density-connected con un punto q rispetto a e MinPts se esiste un punto o che sia density-reachable da entrambi:

- **density-based cluster:** dato un dataset D, un density-based cluster C è un sottoinsieme non vuoto di D che soddisfa le seguenti condizioni:

  - **massimalità**:

    $$\forall p, q \ : \text{ se } p \in C \text{ e } q \text{ è density-reachable da } p \Rightarrow q \in C$$

  - **connettività**:

    $$\forall p, q \in C \ :| \ p \text{ è density-connected a } q$$

  - **rumore**: si definisce rumore l'insieme dei punti che non appartengono ad alcun cluster.

L'algoritmo DBSCAN inizia prendendo un oggetto arbitrario p del

dataset; ne va a calcolare E-neighborhood: se al suo interno vi sono piu di MinPts oggetti, p viene etichettato come appartenente a un cluster, altrimenti come rumore. A questo punto si cerca di espandere il cluster appena trovato includendo iterativamente gli oggetti negli E vicinati di quelli già appartenenti al cluster, questo processo continua fino a quando il cluster non possa più essere espanso.
Si passa quindi ad analizzare il successivo oggetto non visitato del dataset.

Subspace Clustering -- spazi ad alta dimensionalità

Il campo di applicazione del subspace clustering [1] è quello dei dataset ad alta dimensionalità ovvero i dataset con un numero alto di attributi. Questi, a differenza dei dataset che si sviluppano su poche dimensioni, aprono nuove problematiche nel campo del clustering , per definizione un cluster è un insieme di oggetti “simili", la similitudine è un parametro dipendente dagli attributi che descrivono gli oggetti.
In un dataset ad alta dimensionalità non sempre tutti gli attributi hanno la stessa importanza: a seconda della realtà che il dataset descrive alcuni attributi possono essere piu importanti di altri, questa distinzione non può essere catturata dagli algoritmi classici di clustering.
il concetto di similarità (intesa come distanza tra oggetti) tende a perdere di significato all'aumentare del numero delle dimensioni, infatti la distanza tra ogni coppia di punti tende a diventare 0 all'aumentare del numero di dimensioni.
Questo problema e definito “curse of dimensionality” , maledizione della dimensionalità.
Il problema di individuazione degli attributi rilevanti ricade nel campo del pre-processamento del dataset.
Per il problema della maledizione della dimensionalità si utilizzano principalmente due tecniche per la riduzione della dimensionalità del dataset:

- trasformazione degli attributi : vengono selezionati gli attributi più importanti e solo su quelli viene effettuato il clustering
- decomposizione del dominio: il dataset viene diviso in tanti piccoli sottoinsiemi facendo in modo cosi che l'algoritmo di clustering venga eseguito su tutte le dimensioni ma tramite un dataset ridotto.

Gli algoritmi di subspace clustering cercano di superare i problemi di cui sopra andando a cercare i cluster candidati in tutti i sottospazi possibili dello spazio iniziale; una volta trovati questi vengono passati a un algoritmo di clustering tradizionale (es. DBSCAN) che trova eventuali cluster nel dataset ristretto.

Vengono introdotte due tecniche di riduzione della dimensionalità per cercare di risolvere questi problemi:

- feature transformation: con questa tecnica si cerca di ridurre la dimensionalità mappando lo spazio originario in uno spazio di più bassa dimensionalità che ne è una trasformazione lineare. Un esempio di feature transformation e la Principal Component Analysis o PCA [2] e la Singular Value Decomposition o SVD [3].
- feature selection: con questa tecnica si riduce la dimensionalità mappando lo spazio originario in uno spazio che e ottenuto selezionandone alcune dimensioni (quelle supposte piu rilevanti).

Queste tecniche di riduzione della dimensionalità presentano tre problemi, in primo luogo mentre gli attributi originari possono avere un qualche significato, le loro trasformazioni/selezioni perdono di signicato e cosi i cluster che vengono trovati non sono intuitivamente comprensibili. In secondo luogo queste tecniche riducono sì la dimensionalità ma il risultato e che un successivo algoritmo di clustering troverà i cluster solo nello spazio ridotto non considerando tutti gli altri possibili sottospazi e questo potrebbe portare a perdere informazioni sugli oggetti che appartengono a più cluster in differenti sottospazi.
Un secondo approccio per trattare i dataset ad altà dimensionalità e il projected clustering. Con questa tecnica non è piu un singolo

sottospazio ad essere considerato ma delle coppie (Ci; Si)(0ik) con Ci i-esimo cluster e Si sottospazio.
in cui esiste Ci. Il problema di questo approccio è che si continua a non riuscire a catturare le informazioni sugli oggetti appartenenti a diversi cluster su differenti sottospazi.

Con il subspace clustering si cerca di risolvere questa problematica attraverso la rilevazione automatica di cluster in tutti i sottospazi dello spazio originale.

Verranno ora presentati alcuni algoritmi di subspace clustering.

Gli approcci al subspace clustering sono generalmente 2 :

- Algoritmi **TOP-DOWN** → trovano un clustering iniziale da cui valutano il sub-spazio di ogni cluster, rieseguendo poi l'operazione al fine di raffinare i dati
- Algoritmi **BOTTOM-UP** → Cercano zone ad altà densità per poche dimensioni e le combinano per ottenere dei cluster, generalmente sono utilizzati algoritmi Grid-Based.

Algoritmi grid-based

CLIQUE[4]

Uno dei primi algoritmi di subspace clustering è stato Clique (CLustering in QUEst) [4]: questo è un algoritmo bottom-up grid-based che usa un metodo in stile “apriori” per navigare tutti i possibili sottospazi.
L'algoritmo procede come segue:

- Ogni dimensione dello spazio viene partizionata in blocchi equi-dimensionati di larghezza “e” (parametro di input dell'algoritmo); questi blocchi sono chiamati unità.

- Viene calcolata la densità di oggetti costruendo degli istogrammi con valore crescente al crescere del numero di istanze in ogni unità e solo le unità con densità che supera una soglia (input dell'algoritmo) vengono mantenute.
- L'algoritmo procede quindi in stile bottom-up: la generazione delle unità k-dimensionali viene fatto a partire dalle unita (k-1)-dimensionali unendo quelle che hanno le prime (k - 2) dimensioni in comune, dalle unità così generate vengono eliminate quelle non dense.
  Considerando una griglia K dimensionale, si considerano adiacenti due celle che condividono K-1 dimensioni , ovvero data una unità P ed un'unità Q abbiamo che esiste un sottoinsieme K-1 dimensionale P' Q' tra P e Q tale che P' = Q'.
- Viene applicato un criterio di "pruning" con il quale si vanno a tagliare quei sottospazi con bassa copertura (frazione di punti nel sottospazio rispetto ai punti totali del dataset), l'algoritmo procede seguendo una tecnica depth-first , e cerca di restuire sempre i cluster con il più alto numero di dimensioni possibili.
- Si procede trovando i cluster, questi sono definiti come insiemi massimali di unità dense connesse. In ogni sottospazio k-dimensionale viene computato un insieme disgiunto di unità connesse k-dimensionali.
- Infine viene generata per ogni insieme la Minimal Cluster Description: ogni insieme computato viene coperto da regioni massimali e successivamente ne viene determinato la copertura minima.

MAFIA[11]

Un algoritmo che è evoluzione di CLIQUE è MAFIA (Merging of Adaptive Finite IntervAls) [11], questo è sempre un algoritmo grid-based ma invece di usare una griglia con unità di larghezza fissata utilizza una griglia adattiva di larghezza variabile per ogni dimensione.

Per MAFIA il concetto di unità densa è dato da un criterio chiamato "cluster dominance factor" : vengono mantenute solo quelle unità che siano volte (con parametro di input) più densamente popolate della media.
Come determinare ciò ?
È preso come parametro di ingresso un parametro A , tale parametro è rappresentativo della Deviazione Standard che l'unità dovrà avere rispetto alla media per essere considerata densa.
Il secondo parametro è B ed è rappresentativo del margine di differenza tra le varie unità dimensionali per essere collegate.

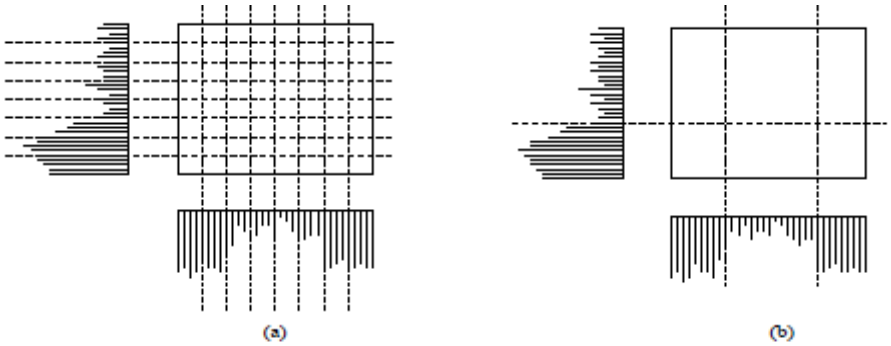
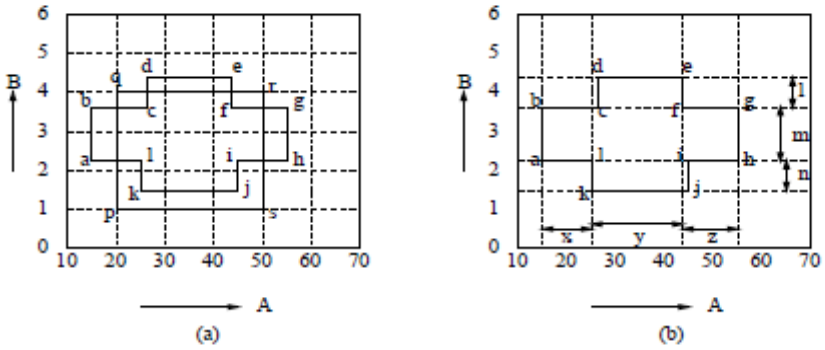


Figure 1. *(a) Uniform grid size (b) Adaptive grid size*

Al momento della divisione delle unità in tutte le proprie dimensioni viene creata la griglia adattativa.
La procedura inizialmente è analoga a CLIQUE , che utilizza una strategia "uniform grid size", con la differenza che mentre in CLIQUE il numero di Unità per dimensione è un parametro di ingresso e quindi "fisso" , in MAFIA il numero di pre-unità è calcolato come MAX(1000, m – n) con m ed n i margini della dimensione , poi però non vengono determinate come unità dense le sole che hanno almeno A deviazione standard, ma le unità tra di loro vengono unite se sono "sufficentemente" simili , ovvero viene vista una similarità nella distribuzione dei dati negli istogrammi .
Quindi se un istogrammi è , con il suo adiacente, simile al massimo con B scarto, i due istogrammi vengono uniti in uno solo.
Questo comporta, dal punto di vista della complessità computazionale , un enorme vantaggio all'algoritmo , difatti si riduce drasticamente il numero di unità dense che l'algoritmo dovrà cercare poi di unire, dato che la complessità di Clique è Esponenziale al numero delle dimensioni trovate all'unità densa.

	Data Dim	Data Size	Cluster Dim	Number of Clusters	Threshold %	Speedup
Data Set A	20	8.85 Million	5	5	2	133.8
Data Set B	30	0.65 Million	8	1	7	66.82
Data Set C	10	0.25 Million	5	3	2	82.13

Si nota come lo speed-up di mafia sia incrementale all'ingrandirsi del dataset.
Difatti con dataset di milioni di dati il fattore esponenziale di CLIQUE pesa enormemente in termini di computazione, e ridurre il numero di unità dense diventa essenziale.
Un altro vantaggio introdotto dall'utilizzo di una griglia adattativa è sui margini del cluster, difatti i margini essendo “mobili” , riescono



. *(a) Cluster discovered by CLIQUE (b) Cluster discovered by MAFIA*

ad identificare meglio i margini dei cluster stessi.

Un'altra importante differenza tra CLIQUE e MAFIA è che quest'ultimo nella sua strategia bottom-up trova le unita k-dimensionali partendo da quelle (k - 1) dimensionali unendo non solo quelle che hanno le prime (k -2) dimensioni in comune ma tutte quelle che hanno qualunque (k - 2) dimensioni in comune. Questo porta ad avere un numero di unità k-dimensionali generate molto più alto .
Questo porta ad un vantaggio, ovvero quello di testare tutte le possibili “forme” che il cluster può assumere migliorando la qualità del cluster.
Il fatto che le unità K dimensionali siano in numero teoricamente più elevato è però soppesato fortemente dall'utilizzo della griglia adattativa che riduce drasticamente in numero di unità dense trovate.

Candidate Dense Units generated by MAFIA and CLIQUE

Dimension		2	3	4	5	6	7	8
MAFIA	Ncdu	21	35	35	21	7	1	0
	Ndu	21	35	35	21	7	1	0
CLIQUE	Ncdu	2313	5739	19215	38484	42836	24804	5820
	Ndu	535	1572	3337	3870	2312	546	0

Successivamente, come in CLIQUE, le unità dense vicine vengono unite per formare i cluster e i cluster ridondanti (rispetto a cluster trovati in uno spazio a più
alta dimensionalità) vengono tagliati.
Una grande limitazione di questi algoritmi è data proprio dall'utilizzo di una griglia multi-dimensionale: i cluster sono connotati all'interno delle celle della griglia. Ad esempio unità k-dimensionali vicine che singolarmente non sono dense potrebbero diventare unità dense se unite, svelando un eventuale cluster. Gli algoritmi presentati ora non tengono in considerazione questa eventualità con la possibilità quindi di non trovare cluster presenti nel dataset.

Algoritmi density-based

SUBCLU

L'algoritmo SUBCLU [5] supera le limitazioni di cui sopra eliminando il concetto di griglia, andando invece a utilizzare il concetto di regione densa e di cluster utilizzato da DBSCAN.
SUBCLU, similmente agli algoritmi presentati, usa una strategia bottom-up per generare i sottospazi. Inizia applicando DBSCAN allo scopo di trovare i cluster in tutti gli spazi 1-dimensionali. Successivamente per ogni cluster generato viene controllato se sia totalmente (o parzialmente) coperto da qualche altro cluster in un sovraspazio dello spazio attuale. A questo punto, per ogni sottospazio k-dimensionale . k, cerchiamo tutti gli altri sottospazi aventi (k - 1) attributi in comune e li uniamo per generare i sottospazi (k + 1) dimensionali.
Per la proprietà di monotonicità, per ogni sottospazio cosi generato

deve contenere ogni sottospazio k-dimensionale precedente.
Possiamo quindi tagliare quei candidati (k + 1) dimensionali che hanno almeno un sottospazio k-dimensionale non incluso in Sk.
Nell'ultimo passo l'algoritmo va a generare i cluster (k + 1)-dimensionali e i corrispondenti sottospazi (k + 1)-dimensionali che contengono questi cluster, cioè per ogni sottospazio S (k + 1)-dimensionale si va a prendere il sottospazio k-dimensionale T tale che T S e su ogni cluster in T viene eseguito DBSCAN.
Per minimizzare il costo di DBSCAN si sceglie il sottospazio Tb che contenga il numero minore di oggetti nei cluster in esso contenuti.
con s sottospazio, Ci i-esimo cluster di s e Cs insieme dei cluster di s.
Grazie a questa euristica vengono minimizzate il numero di computazioni del vicinato effettuate da DBSCAN riducendo di molto il costo computazionale complessivo dell'algortimo.
Riassumendo, da una parte gli algoritmi grid-based mostrano un tempo di esecuzione basso ma al costo di una precisione nell'individuazione dei cluster bassa (dovuta alle euristiche introdotte e al posizionamento della griglia), dall'altra parte gli algoritmi density-based mostrano una precisione notevolmente più alta ma al costo di un tempo di esecuzione molto elevato dovuto alle continue computazioni dei vicinati (necessarie per il calcolo della densità).
Un algoritmo di subspace clustering necessita di esibire un compromesso tra
precisione e tempo di esecuzione,INSCY, un algoritmo di subspace clustering density-based che grazie a una struttura adhoc di indicizzazione dei sottospazi e ad una euristica di pruning degli stessi permette un abbassamento notevole dei tempi di esecuzione pur mantenendo un'ottima precisione.

INSCY

INSCY [6] nasce da due necessità:

- trovare tutti e i soli cluster non ridondanti in tutti i sottospazi

- mantenere un tempo di esecuzione basso

Queste due necessità vengono risolte attraverso una strategia di visita del reticolo dei sottospazi di tipo depth-first: per ogni regione viene effettuato il mining di tutti i cluster in tutti i sottospazi prima di passare alla regione successiva.
Questo porta due vantaggi principali, effettuare dapprima il mining degli spazi ad alta dimensionalità permette di tagliare tutti i sottospazi ridondanti; in secondo luogo con questa strategia di visita e possibile un'efficiente indicizzazione di tutti i possibili sottospazi.
INSCY utilizza per la rappresentazione del dataset una griglia a conservazione di densità per la discretizzazione dello spazio unita a una struttura dati ad-hoc per l'indicizzazione dello stesso: lo SCY-tree.

Una griglia a conservazione di densità e una griglia cosi come definita dagli algoritmi di clustering grid-based (ovvero una suddivisione di ogni dimensione in intervalli di larghezza ssata) con in più l'aggiunta dei bordi di connettivita ovvero regioni di larghezza posizionate nel bordo di ogni intervallo.

DOC[10]

Doc è un semplice algoritmo Density-based,
Un cluster C proiettato è definito da un set di punti in C, e un set di dimensioni rilevanti.
In aggiunta, tre parametri (alfa , beta e omega) .

- Omega : controlla l'estensione del cluster, la distanza tra i record all'interno della stesso cluster è dato da omega e alpha .
- Alpha rappresenta la densità minima minima del cluster trovato , ogni cluster deve contenere almeno alpha * S punti , dove S è la size del database.
- Beta invece riflette l'importanza della grandezza del sottospazio .

DOC trova un cluster alla volta, per ogni step , prende un punto Random P dal database S e tenta di trovare un cluster centrato in P . Per far ciò , ricorre ad un loop interno che seleziona un set di campioni X inclusi in S ed un set di dimensioni D , dove tutti i punti in X hanno distanza omega da p .
Quindi , un cluster C di X è approssimato da un "box" di larghezza 2omega attorno a p nelle dimensioni rilevanti.
C è definito da un set di punti di S nella tale box . Il processo è poi ripetuto per un numero random di punti p e campioni X per ogni p .
Tra tutti i C trovati , il cluster con qualità maggiore è infine selzionato . La qualità del cluster C è definita da U(a,b) = alpha (1/beta) alla b. Dove alpha è il numero di punti in C e b è la dimensionalità di C .
Dopo che un cluster è stato scoperto , i record inclusi in esso sono rimossi dal campionamento e il processo è iterativamente applicato al resto dei punti.
Con tale approccio il numero dei cluster è trovato automaticamente , ma si trovano sono cluster piccoli e richiede un elevato tempo di esecuzione per l'individuazione di cluster con elevata accuratezza.

MINECLUS [11]

MineClus è un algoritmo che migliora DOC sia dal punto di vista della qualità dei cluster che dal punto di vista delle performance .
Esso è formato da 4 fasi :

- Fase di iterazione : questa fase è attua a trovare i cluster C come accade per DOC utilizzando un algoritmo ottimizzato uGrowth che va a sostiuirsi all'inner loop randomizzato di DOC per una più veloce convergenza di tale algoritmo.
  Anche in questo caso viene generato un cluster alla volta , è possibile che il cluster risultante sia una parte di uno più grande (grande oltre i limiti del box) . Utilizzando la "manattan segmental distance" vengono assegnati anche record che hanno al massimo max_dist distanza dal centroide .
- Fase di pruning : nella fase di "potatura", i cluster con u

valori significativi minori degli altri vengono eliminati , viene trovata una posizione wS tale che rappresenti un valore mediano i quali cluster con indice minore di wS siano cluster forti e quelli con indicie maggiore siano cluster deboli . I cluster deboli sono tagliati.

- Fase di merge : questa fase è applicata solo se l'utente vuole un numero k di cluster , in questo caso i cluster forti sono mergeati fino a quando non ne rimangono K .
- Fase di raffinamento : similmente a PROCLUS si migliorano i cluster assegnando i rimanenti records nel dataset ai cluster.

Algoritmi Top-Down

Proclus [7]

Proclus è stato il primo algoritmo di subspace cluster di tipo Top-Down .
É simile a CLARANS [8] , esegue un campionamento dei dati , poi seleziona un set di K mediani ed iterativamente migliora il clustering. L'algoritmo consiste in un approccio a tre fasi che sono.

- Inizializzazione → utilizza un algoritmo “greedy” per selezionare un set di potenziali mediani abbastanza lontani tra loro, l'obbiettivo è quello che ogni cluster sia rappresentato da almeno un'istanza nel set selezionato
- iterazione → seleziona un set random di K metoids dal dataset ridotto, e sostituisce i mediani peggiori con nuovi presi random , controllando che il clustering sia poi migliorato; la qualità del clustering in questo è basata sulla distanza media tra istanze e il mediani più vicino .
  Per ogni mediani , viene selezionato un set di dimensioni con una bassa distanza (rispetto all'aspettativa statistica) . Il numero totale di dimensioni associate ai mediani deve essere K*L , dove L è un parametro di input rappresentativo della dimensionalità media del sotto-spazi di cluster. Dal momento

in cui i sottospazi sono poi selezionati per ogni mediani, viene utilizzata la "average Manhattan segmental distance" per assegnare i punti ai mediani, formando i cluster

- raffinamento clustering.
  La fase di raffinamento computa nuove dimensioni per ogni mediani basato sui cluster formati per il reassignamento dei punti nei mediani, rimuovendo il rumore (outliers) .

PROCLUS è più veloce di CLIQUE per dataset di grandi dimensioni , e i cluster sono istanze con mediani associati e sotto-spazi che formano partizioni non sovrapposte del dataset.
L'accuratezza di PROCLUS è molto sensibile ai parametri di input , i quali possono essere anche difficili da determinare .

FINDIT

Dimension – Oriented Distance.

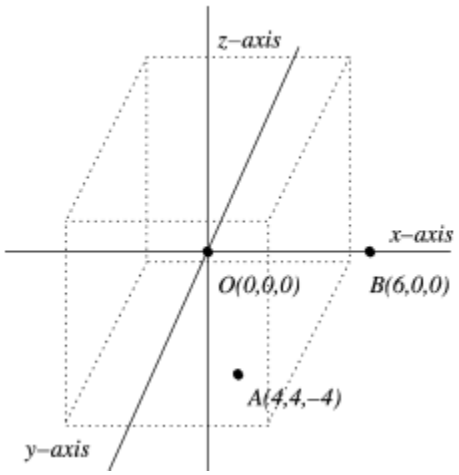
L'unica misura per le distanze è chiamata DOD ; utilizza il concetto di differenza di dimensione (dimensione x rispetto a y) e differenza di valore assieme; Si misura quindi la prossimità di due punti attraverso un dato espilon ; se la differenza è inferiore all'epsilon dato i due punti sono considerati "sufficentemente" vicini.
La filosofia che sta alla base di questa strategia è che nell'utilizzo di un elevato numero di dimensioni è più importante essere vicino in molte dimensioni piuttosto che essere più prossimi ma in meno dimensioni.

Per un punto p in un dataset Considerando D un sottospazio composto nelle dimensioni ove p ha dei valori dimensionali significativi (ovvero le dimensioni sulle quali si vuole controllare che il punto p sia nel cluster) .
Definendo p(d) il valore di p nella dimensione d e q un altro punto, la DOD si ha :

$$dod_\epsilon(p \rightarrow q) = |D_p| - |\{d | |p(d) - q(d)| \leq \epsilon, d \in D_p \cap D_q\}|,$$

Per spiegare bene il concetto facciamo un esempio.

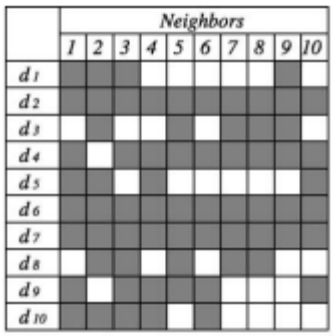
Considerato epsilon = 5 otteniamo , accanto al punto O :



Utilizzando una normale distanza Euclidea o la Manhattan Distance otteremmo che il punto più vicino a O è B .
Nel concetto di DOD invece si tiene conto dell'ipercubo circondante O e di lunghezza epsilon per ogni dimensione; in questo caso notiamo come A ricada all'interno di tale cubo mentre B no .

Criterio di selezione delle dimensioni

FindIT strima le dimensioni significative (quelle che formeranno il cluster) di un punto p partendo dallo studio del vicinato .
Parametri necessari per l'individuazione dei cluster sono quindi la epsilon di "distanza" e la soglia del numero di dimensioni corrispondenti .
Dato il punto p controllo il vicinato , ad esempio prendo i 10 punti del suo vicinato , data questa informazione controllo per ogni dimensione di p se il vicinato n-esimo ricade all'interno della epsilon per la dimensione voluta... ponendo soglia uguale a 7 ottengo (considerando in grigio le dimensioni che ricadono nella soglia) che tale punto appartiene ad un cluster sulle dimensioni 2,4,6,7 .



(b) Point p's 10 nearest neighbors

Algoritmo

I parametri di input dell'algoritmo sono Dmindist e Cminsize ; CminSize identifica il numero minimo di dimensione del cluster, il secondo parametro rappresenta invece la minima distanza intra-cluster , quindi se due cluster restano dotto la distanza min-dist per un determinato epsilon, essi sono considerati come uno.
Dmindist può essere considerato essere il numero di dimensioni correlate che il cluster dovrebbe avere per essere mergeato.
L'algoritmo FINDIT si compone di tre fasi .

Fase di campionamento :
Vengono effettuati 2 campionamenti differenti random, il primo set S è costruito dal dataset come distribuzione rappresentante il set dato , il secondo secondo è chiamato M ed è utilizzato come mediani del cluster originale
Come avviene il criterio di selezione ?
Il criterio di selezione tenta di ottenere inizialmente un campione di M ed S che rappresenti il più possibile l'intero dataset , per far ciò si utilizzano i chernoffBounds [22] ,
Ovvero considerando Xj una variabile random che ha valore 1 . se il punto appartiene al cluster e 0 se non vi appartiene, possiamo assumere che ogni X relativa ad ogni punto è una variabile indipendente dall'altra.
Il chernoff bound ci permette di individuare il numero minimo delle istanze da campionare in S ; assumiamo che ogni cluster in s debba contenere almeno E punti con una probabilità 1-d , il valore di s è

computato da :

$$= \xi k\rho + k\rho \log\left(\frac{1}{\delta}\right) + k\rho\sqrt{\left(\log\left(\frac{1}{\delta}\right)\right)^2 + 2\xi \log\left(\frac{1}{\delta}\right)}.$$

Considerando K costante il numero di cluster , e p è un valore dato da $minsize = N/k\rho \ (\rho \geq 1)$, con minsize = minima dimensione del cluster (parametro di input). K uguale ad N/Cminsize .
Settando poi E = 30 , e d=0,01 , otteniamo che anche il più piccolo cluster presente nel dataSet (rappresentato da CminSize) ha il 99% di probabilità di avere 30 punti che lo rappresentano in S.
Quindi con N = 100.000 abbiamo una grandezza di s di circa 1037 istanze .
Anche M viene estrapolato secondo lo stesso criterio, ma in questo caso E è posto ad 1 in quanto solo un centroide è da attribuire ad M.
Sempre nel caso di N=100.000 M ha una grandezza di 224.
Una volta impostata la grandezza di S ed M , le istanze sono prelevate Random.

Fase di formazione cluster
Lo scopo di questa fase è ottenere i cluster originali con le relative dimensioni correlate attraverso il metodo di dimension voting, quindi i mediani vicini tra loro saranno raggruppati
In tale fase viene definita espilon , viene ripetuta diverse volte con un epsilon crescente , nel dettaglio 25 volte con valori compresi tra (val/100 , val /4) con val il range di valori della data dimensione. Tali set ottenuti sono poi valutati e viene scelto uno.
Per definire le dimensioni correlate di tutti i punti in M vengono selezionati da S un numero V di "Votanti" , per ogni dimensione di M per definire se tale dimensione è una dimensione del cluster ogni V controlla che la d-iesima dimensione di M ricada all'interno dell'espilon di V , in caso affermativo vota "si" .
Se i si superano la soglia allora tale dimensione è considerata parte del cluster.
Qual'è un buon numero di per V ? e una soglia per i si ?
Per i si si considera una probabilità di distribuzione binomiale B(n: V,p) con p = 2e/val .

Fase di assegnazione
Una volta stabiliti i mediani ogni punto sarà preso e inserito nel cluster più vicino a lui, se non ricade nell'influenza di nessuno dei cluster è considerato rumore e scartato.

Algoritmi per grandi moli di dati.

BIRCH

(balanced Iterative reducing and clustering using hierarchies) è un algoritmo gerarchico specializzato per operare con grandi moli di dati. Un vantaggio dell'utilizzo di Birch è la sua abilità di clusterizzare incrementalmente e dinamicamente punti multi-dimensionali nel tentativo di produrre i migliori cluster possibili.
Birch richiede un solo scan del database nel caso comune , è anche il primo algoritmo che gestisce il "rumore" (punti che non fanno partedi un determinato pattern) .
È un algoritmo locale in quanto ogni decisione riguardante il clustering è effettuate senza dover scannerizzare tutte le istanze e i cluster correnti. Si basa sul concetto che lo spazio non è mai uniformemente occupato e non tutti i punti sono equamente importanti,
Minimizza le operazioni di I/O ma ha un alto consumo di memoria nella ricerca dei sotto-cluster .
Birch è incrementale in quanto non richiede l'intero data set in anticipo.
Algoritmo:
Dato un set di N d-dimensionali istanze , la "Clustering Feature CF" di tale set di dati è definito dalla tripla CF = (LS + SS) dove LS è la somma lineare ed SS è il somma quadratica dei punti .
Le Clustering features sono organizzate un albero detto CF tree , il quale è un albero bilanciato in altezza con due parametri : il Brancing factor B e una soglia T .
Ogni nodo non foglia del CF contiene al massimo B entries dal

“form” [CF_i,child_i], dove I è un puntatore al suo nodo figlio è Cfi è la clustering feature che rappresenta il sotto-cluster associato.
Un nodo foglia contiene al massimo i entries ognuna nella forma Cfi , ha anche due puntatori prev e next i quali sono utilizzati per concatenare assieme i nodi foglia.
La size dell'albero dipende da un parametro T .
Un nodo è inserito in una pagina di Taglia P , B ed L sono determinati da P.
Quindi P può variare per effettuare un tuning delle performance . La rappresentazione è veramente compatta perchè poi ogni nodo foglia non è una singola istanza bensi un sotto-cluster. L'algoritmo come primo step scandisce tutti i dati e costruisce un CF tree iniziale utilizzando il dato limite di memoria .
Il secondo step scandisce tutte le foglie del CF tree iniziale e lo ricostruisce creandone uno più piccolo ed allo stesso tempo rimuove gli outliers e raggruppa i sotto-cluster ottenuti in altri sotto-cluster più grandi.
Nel terzo step utilizza un algoritmo di clustering esistente per effettuale il cluster di ogni foglia, poi un algoritmo di clustering gerarchico ed agglomeratico è utilizzato direttamente sui sotto-cluster rappresenta dai vettori delle CF .

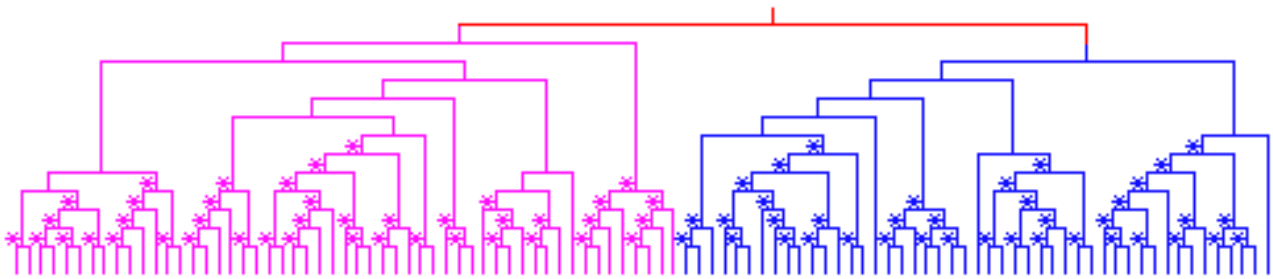
Grin Algorithm

Il GRIN prende le sue basi dal birch , applicando la legge di gravitazione universale ai cluster per ottenere un metodo più veloce del Birch. Tale algoritmo è O(n) come complessità computazionale.
Partendo subito dai difetti si può dire che GRIN in alcuni casi non soddisfa la qualità dei cluster e
trova il suo “tallone d'achille” nella configurazione dei parametri che possono portare da una buona qualità ad una pessima qualità dei cluster trovati.
Grin è un algoritmo incrementale , ed anche la sua complessità computazionale può variare, soprattutto nel caso in cui i dati successivi alle prime istanze arrivano spazialmente molto distanti da

i precedenti , nell'ipotetico “worst case” tale complessità può raggiungere O(n^2).

IL GRIN lavora in due fasi . In ognuna di esse invoca un algoritmo di clustering gerarchico agglomerativo ;
L'idea chiave dietro al GRIN è qualsiasi cluster anche di forma irregolare può essere rappresentato da un'insieme di sfere.
Il clustering a sfere rappresenta un clustering primario , le sfere tra loro possono intersecarsi.
Il centroide di un cluster è definito essere il centro geometrico del cluster ; il raggio rappresenta la massima distanza tra il centroide l'istanza più lontana.
Nel GRIN è assunto che tutte le istanze sono prima bufferate in un Incoming Data Pool ; il GRIN è immune all'ordine in cui arrivano tutti i dati.
Viene costruito un dendrogramma dei dati , dove le foglie finali sono le istanze ed i nodi rappresentano i cluster; tali cluster hanno forma sferica e devono rispettare 2 condizioni.

La prima è che contengano almeno Min valori (con Min parametro).
La seconda è che passi un test statistico come segue:
Si ipotizza che il cluster sia una sfera e che i dati siano uniformemente distribuiti all'interno di essa ; a seguito di ciò si effettua un chi quadro test per determinare la distribuzione delle istanze .

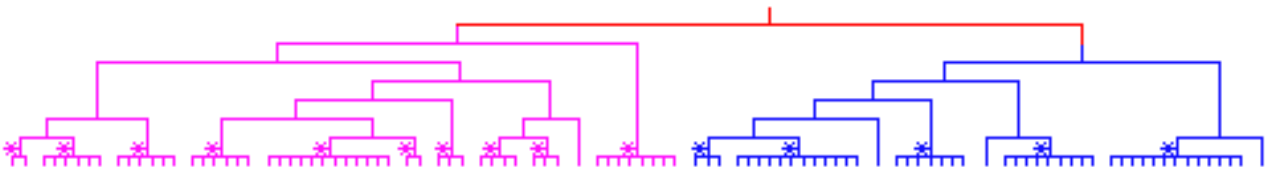


I cluster che passano tale test vengono marcati con * , i cluster che contengono una singola istanza non sono applicabili a tale test e

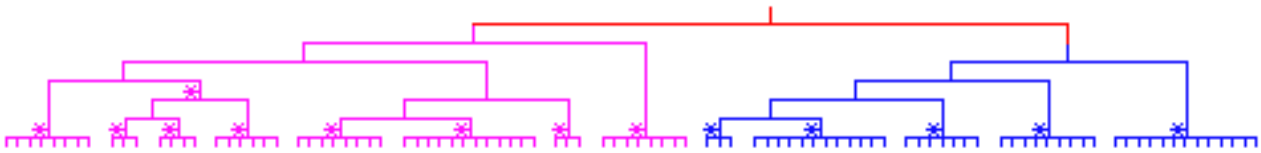
vengono considerati come cluster sferici primitivi.
A questo punto vi è la fase di raggruppamento / potatura, un clustering sferico diventa un cluster foglia se:

- Il cluster ha forma sferica (*)
- Il cluster ha dei discententi e tutti i suoi discendenti hanno forma sferica.
- Il suo padre non soddisfa ne la condizione 1 ne la 2

Otteniamo



Un ultimo step che si può applicare è quello di rimuovere i punti "fuori" (outliers) ed inserirli in un outliers buffer.. tali punti poi potrebbero formare dei cluster successivamente con l'inserimento di istanze successive.
Ragionando sulla densità in pratica si ha che se un cluster cluster non raggiunge una densità di soglia è considerato outliers e quindi viene scartato.
A seguito della "messa a parte" degli outliers si ricarica il dendrogramma , questo perchè la presenza di tali "semi cluster" incide sulla qualità dei cluster stessi.
Il dendrogramma ricostruito finale risulta quindi :



- Ogni cluster è formato da : il centroide → il centro geometrico del cluster
- 2 il raggio → la massima distanza tra il centroide e il dato più lontano
- 3 la massa del cluster → il numero di istanze contenute nel cluster

Nella seconda fase dell'algoritmo ogni istanza dell'incoming data pool viene esaminata , se una di esse finisce all'interno del raggio di un cluster foglia tale istanza è aggiunta al cluster.
Se l'istanza finisce nell'intersezione tra 2 foglie , allora vince la foglia con gravità maggiore.
Se non finisce in nessuna foglia è considerata outlier, inserendola negli otulier si può tentare di formare un nuovo cluster con quest'ultima tra tutti gli outlier.
Quando il numero degli outlier eccede una quantità massima , il clustering viene invocato e costruito un nuovo dendrogramma ; in questo procedimento di ricostruzione gli oggetti primitivi sono i cluster foglia e le istanze nel buffer outlier.
Rifatto il dendrogramma si ripercorrono gli stessi passi effettuati in precedenza. A seguito di questo secondo tentativo il buffer di outlier potrebbe essere ancora pieno (non ho fatto altri cluster) quindi considero i punti rumore e svuoto il buffer.

GENIC

Genic è un algoritmo generalizzato di clustering incrementale , l'approccio utilizzato è quello di dividere lo stream delle istanze in finestre . L'algoritmo si compone di 5 fasi :

- Selezione dei parametri → si fissa il numero di k centri , il numero di punti iniziali m e la taglia della generazione n
- Inizializzazione → si selezionano da m punti candidati come centri iniziali e si assegna un peso = 1 ad ogni centro.
- Per ogni conseguente punto p si
    - Incrementa il contatore
    - Si ricerca il centro candidato più vicino al punto p

- Si sposta il centro candidato utilizzando la seguente formula

$$c_i = \frac{(w_i * c_i + p)}{(w_i + 1)}$$

- si incrementa il peso del centro candidato di 1 .
- riesegue il punto 3 fino a quando il contatore non è uguale ad N.

- Miglioramento dei cluster .
  - Si calcola la probabilità di sopravvivenza , ovvero l'incidenza del centro rispetto a tutti i centri trovati (quello che ha più size sarà quello con p più alta).
  - Si releziona un numero casuale compreso tra 0 e 1 e se tale numero è maggiore di Pi il centro viene scartato e si seleziona un nuovo punto random , rieffettuando il punto 3.
- Calcolo dei cluster finali .
  - Finita la fase di miglioramento si ottengono m cluster , i quali vengono raggruppati al fine di trovare i k cluster finali.

IHC

IHC è un algoritmo gerarchico agglomerativo ,rappresenta un trade-off tra gli approcci locali (bottom-up) e globali (top -down) , anch'esso è un algoritmo incrementale.
IHC costruisce il concetto di gerarchia tramite due proprietà “omogeneità e monotonicità” .
Un cluster omogeneo è un set di istanze con densità simile. Una gerarchia di cluster che soddisfa la proprietà di monotonicità se la densità del cluster è sempre maggiore a quella del suo “padre” . Quindi la densità incrementa monotonicamente partendo dalla root fino ad arrivare ai nodi foglia.
Come per tutti gli algoritmi gerarchici IHC mantiene una struttura ad albero, dove ogni nodo mantiene 2 informazioni “centro del cluster “ e la densità.

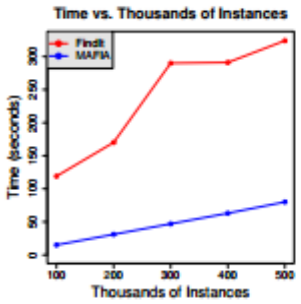
La densità è data dalla distanza media dal nodo più vicino , ed un modo facile per ottenere tale valore è creare un MST (minimum spanning tre) degli oggetti nel cluster.
L'approccio per incorporare un nuovo punto nella gerarchia è diviso in due step.

- Trovare il punto più vicino ad un nodo foglia.
- Partendo dal padre dell'ultimo nodo foglia , eseguire una “upward search” (per localizzare il cluster che possa contenere il nuovo punto con il minore cambiamento di densità e meno incidenza sulla monotonicità del cluster.

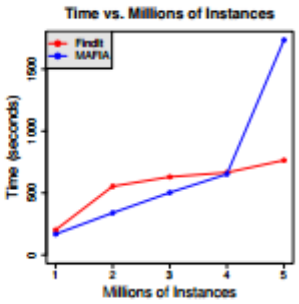
Considerazioni e paragoni tra algoritmi.

Mafia vs FindIt.

Ponendo a confronto un algoritmo bottom-up ed uno top-down di sub-space clustering grazie al lavoro di H.Liu[1] , possiamo trarre le seguenti consideazioni :

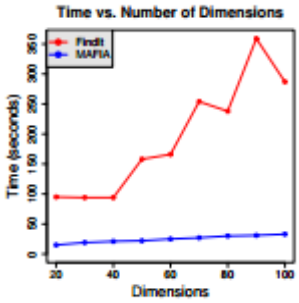


(a) Thousands



(b) Millions

- Mafia ,a parità di dimensioni, in caso di basso – medio numero di istanze (fino a 4.000.000) è più performante di FindIt specialmente per i valori sotto al milione.
- FindIt a parità di dimensioni, mantiene invece una costanza maggiore per valori con enormi quantità di dati (più di 4 milioni) , mentre Mafia trova una critica impennata alle performance.



- Mafia , a parità di istanze, risulta molto più stabile e lineare relativamente all'incremento delle dimensioni del dataset.

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(1, 8, 9, 15, 18)	(7, 9, 18, 20)	(12, 15, 18, 19)	(5, 14, 18, 19)

- Mafia Con un dataset di 100.000 istanze e 20 dimensioni , risulta individuare bene tutti i cluster, a parte di una dimensione per i cluster 1,3,4 e 5.

Cluster	1	2	3	4	5
Input	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)
Output	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)

- FindIt copre bene tutti i cluster per un dataset analogo con stesso numero di istanze e stesso numero di dimensioni.

Cluster	1	2	3	4	5
Input	(1, 5, 16, 20, 27, 58)	(1, 8, 46, 58)	(8, 17, 18, 37, 46, 58, 75)	(14, 17, 77)	(17, 26, 41, 77)
Output	(5, 16, 20, 27, 58, 81)	*None Found*	(8, 17, 18, 37, 46, 58, 75)	(17, 77)	(41)

- Aumentando il numero di Dimensioni (100) FindIt “perde colpi” , in particolare non riesce a trovare il cluster 2 , il cluster 1 è sbagliato ed i cluster 4 e 5 sono incompleti.

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(8, 9, 15, 18) (1, 8, 9, 18) (1, 8, 9, 15)	(7, 9, 18, 20)	(12, 15, 18, 19)	(5, 14, 18, 19)

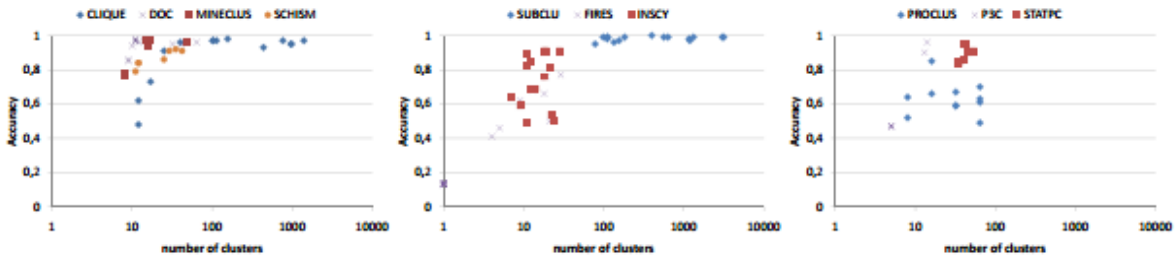
- Mafia invece risulta più stabile , si perde ancora delle dimensioni ed il cluster 2 viene suddiviso in 3 sotto-cluster , ma mantiene più fede ai cluster reali.

CONCLUSIONI.

- In caso di dataset ad alto numero di istanze e basso numero di dimensioni è più conveniente utilizzare FindIt come algoritmo di subspace Clustering.
- In caso di dataset ad alto numero di dimensioni e meno numero di istanze è più conveniente utilizzare Mafia.
- Entrambi gli algoritmi non “brillano” per qualità dei cluster, in particolare Mafia dipende molto dal posizionamento della griglia.

Confronto tra algoritmi ibridi , e density-based.

Grazie a [17] abbiamo una migliore panoramica degli algoritmi analizzati e del loro comportamento .

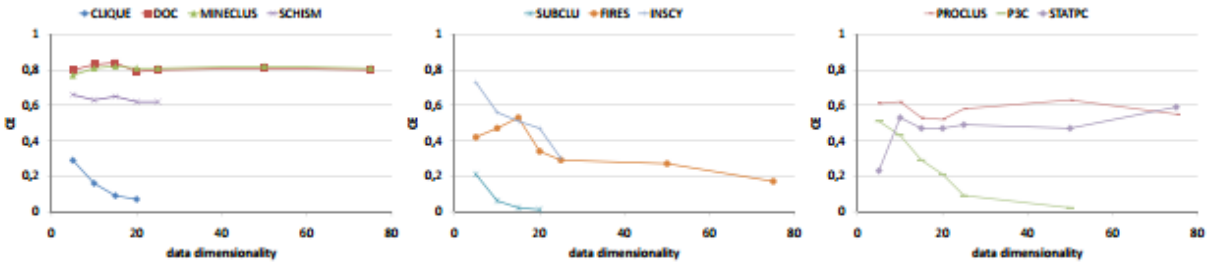


(a) cell-based (b) density-based (c) clustering-oriented

Figure 2: Accuracy measure vs. number of clusters

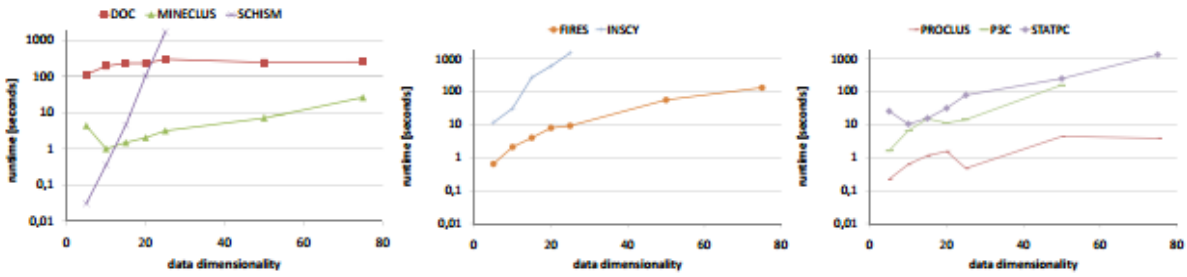
Notiamo che :

- Clique , sub-clu e proclus per raggiungere elevate accuratezze devono suddividere il sottospazio in un elevato numero di cluster, rendendo l'informazione più frammentaria e quindi una qualità di cluster inferiore.
- MINECLUS ed INSCY invece tendono ad un ottima qualità utilizzando un basso numero di cluster, manifestando una migliore qualità.



(a) cell-based (b) density-based (c) clustering-oriented

- MineClus ottiene sempre un'ottima qualità dei cluster rispettivamente all'incremento delle dimensioni , PROCLUS seppur mantiene sempre una qualità bassa (< 0,6) si mantiene costante all'aumento di dimensione .



(a) cell-based (b) density-based (c) clustering-oriented

Figure 9: Scalability: runtime vs. database dimensionality

- Parlando di tempistiche si nota come il più performante è

sicuramente PROCLUS , Mineclus rimane su tempi accettabili mentre INSCY accusa notevolmente l'aumento di dimensioni .

Pendigits (size: 7494; dim: 16)

	F1		Accuracy		CE		RNIA		Entropy		Coverage		NumClusters		AvgDim		Runtime	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0,30	0,17	0,96	0,86	0,06	0,01	0,20	0,06	0,41	0,26	1,00	1,00	1890	36	3,1	1,5	67891	219
DOC	0,52	0,52	0,54	0,54	0,18	0,18	0,35	0,35	0,53	0,53	0,91	0,91	15	15	5,5	5,5	178358	178358
MINECLUS	0,87	0,87	0,86	0,86	0,48	0,48	0,89	0,89	0,82	0,82	1,00	1,00	64	64	12,1	12,1	780167	692651
SCHISM	0,45	0,26	0,93	0,71	0,05	0,01	0,30	0,08	0,50	0,45	1,00	0,93	1092	290	10,1	3,4	5E+08	21266
SUBCLU	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
FIRES	0,45	0,45	0,73	0,73	0,09	0,09	0,33	0,33	0,31	0,31	0,94	0,94	27	27	2,5	2,5	169999	169999
INSCY	0,65	0,48	0,78	0,68	0,07	0,07	0,30	0,28	0,77	0,69	0,91	0,82	262	106	5,3	4,6	2E+06	1E+06
PROCLUS	0,78	0,73	0,74	0,73	0,31	0,27	0,64	0,45	0,90	0,71	0,90	0,74	37	17	14,0	8,0	6045	4250
P3C	0,74	0,74	0,72	0,72	0,28	0,28	0,58	0,58	0,76	0,76	0,90	0,90	31	31	9,0	9,0	2E+06	2E+06
STATPC	0,91	0,32	0,92	0,10	0,09	0,00	0,67	0,11	1,00	0,53	0,99	0,84	4109	56	16,0	16,0	5E+07	3E+06

- Confrontando un'applicazione su un Dataset Reale composto da 16 dimensioni e circa 8000 istanze si nota la vera debolezza di MineClus, ovvero il tempo di esecuzione che risulta molto alto, mentre algoritmi quali CLIQUE e PROCLUS sono decisamente più veloci.

CONCLUSIONI

In ottica di clustering di grandi moli di dati le problematiche ricorrenti sono :

- Qualità dei cluster trovati
- Tempi computaziali elevati
- Utilizzo delle risorse (RAM , disco)
- Maledizione della dimensionalità
- Taratura dei parametri.

Di questi punti gli algoritmi di sub-space clustering tentano di risolvere soprattuto i punti 2,3 e 4 .
Gli algoritmi density – based sono invece più concentrati sul punto 1 .
Abbiamo visto anche casi ibridi cone INSCY , ma che non hanno dato particolari risultati .

Generalizzando abbiamo che :

- Gli algoritmi di sub-space clustering offrono una buona soluzione alla maledizione della dimensionalità , MAFIA e FINDIT utilizzando uno l'incrementalità l'altro un campionamento fanno bene fronte al consumo di risorse , con tempi computazionali ridotti ed adatti a lavorare con milioni di dati .
- Gli algoritmi density-based non sono adatti a dataset di ordini superiori alle centinaia di migliaia di record, stante la proprià natura tendenzialmente esponenziale di complessità , ma offrono un'ottima qualità nei cluster.
- La taratura dei parametri è una problematica comune, ed il parametro più comune a tutti gli algoritmi è il concetto di "soglia" per il quale si accetta un cluster o meno.

Statistica applicata al clustering – Hopking

La statistica di Hopking [21] pone le basi allo studio di tesi seguente; tale studio afferma che un data privo di cluster inerenti alla propria struttura dati può essere rappresentato come una una serie casuale di punti ed attributi , tale da non dover essere partizionata alla ricerca di un cluster.
Lo studio di "tendenza al Clustering" non da informazioni su quanti cluster vi sono all'interno di un cluster , fornisce solamente informazioni sulla presenza naturale di raggruppamenti di dati ; successivamente sarà quello che verra Definito NON CASUALE + .
Lo studio fornisce un concetto "statistico" nel campo della tendenza al clustering e mostra come applicarlo al processo di validazione del clustering effettuato dal partizionamento dei dataset, ovvero basandosi sulla "casualità dispazio" .
La distribuzione di un processo Poissoniano è presa come ipotesi 0 , ovvero una distribuzione dei punti del dataset casuale , alla quale poi si applicano i test statistici di confronto quali Hopkings [22],

Holgate[23] , T-square [24] , Eberhardt [25] .
La statistica di hopkings è facile da utilizzare e comprendere ed è comparabile a livello di affidabilità alla statistica di Holgate.
Sia *X={x, 1 i=l to n}* una collezione di *n* "pattern" in un *d* spazio dimensionale tale che **s,** = *{.xlI.* -U,., .., .v,d).
Inoltre sia Y= *{Cv,* ij=l to *m}* con *m* campionamenti disposti casualmente nello spazio d-dimensionale con m << n .
Una "finestra di campionamento" può essere pensata come un sotto-spazio dell'intero dataset .
Sono definiti due tipi di distanze :

- a , come la minima distanza da v al suo pattern più vicino in X
- w , come la minima distanza tra un pattern casuale in X e il suo vicinato più adiacente.

Questa statistica esegue una comparazione delle distanze del vicinato della distribuzione casuale poissoniana con quelle appartenenti al dataset.

$$H = \frac{\sum_{j=1}^{m} u_j^d}{\sum_{j=1}^{m} u_j^d + \sum_{j=1}^{m} w_j^d}$$

Quindi quando i pattern saranno aggregati o clusterizzati in zone specifiche , la distribuzione delle distanze tra di essi nel caso del dataset sarà più piccola del caso casuale, quindi si può determinare che in quell'area vi è la presenza di UN cluster, ma non di QUALE cluster.
Il ragionamento analogo può essere applicato nelle aree ove la distribuzione delle distanze è minore di quella casuale, identificando quindi aree NON CASUALI non dense (NON CASUALI -) .

I limiti del task di clustering[25]

Organizzare i dati in gruppi sensibili è un attività all'attenzione di

tutti , sia a livello scientifico che di business ; non soprende quindi vedere come l'attenzione rivolta al clustering sia in continuo incremento ; è importante ricordare che il clustering non è altro che un supporto all'esplorazione dei dati, non ci fornirà mai un indicazione esatta, quello che restituisce è solo un'ipotesi con un determinato indice di errore .
E mentre ogni giorno vengono creati nuovi algoritmi , non ve ne è uno solo per il quele è stato dimostrato il dominio di esso su tutti i contesti applicativi e le numerose problematiche ad esso annesse.
Prendendo come esempio un dominio di estrapolazione di conoscenza , dato lo stesso set di documenti differenti “gruppi di utenti” (legale, amministrativo, manageriale ,ecc) potrebbero essere interessati a generare partizioni dei dati a seconda dei loro bisogni ; in metodo di cluster quindi potrebbe essere ottimale per un gruppo (perchè lavora su determinati attributi) potrebbe non esserlo per un'altro.
Le problematiche maggiori , durante uno studio di clustering , sono le seguenti:

- Vi è la necessità di una liberia di dati benchmark sul quale effettuare i test ; tali dati dati dovrebbero essere piuttosto vari (documenti,immagini,time-series,transazioni clienti ,ecc). Tali benchmark dovrebbero inoltre includere dati dinamici e statistici, con attributi qualitativi e quantitativi. Qualcosa è gia reso disponibile da UCI ML e KDD repository.
- Occorre prestare attenzione al dataset utilizzato, soprattutto nell'individuazione degli attributi significativi, ed a seconda dell'applicazione potrebbe essere necessario scendere a più livelli dettaglio nei cluster trovati, ad esempio per certi lati applicativi due cluster tra di loro poco coesi potrebbero essere accettabili anche se uniti in un unico cluster, mentre in altre farebbe la differenza una migliore separazione .
- Occorre prestare un occhio di riguardo alla complessità computazionale dell'algoritmo elaborato , in quanto quello che può anche essere ottimo nel piccolo potrebbe essere ingestibile nel grande.
- Occorre elaborare algoritmi di cluster stabili alla variazione

dei dati , la stabilità e la consistenza alle variazioni dei dataset sono aspetti fondamentali.

- Data la intrinseca difficoltà a svolgere il task di clustering è sensato sviluppare tecniche di clustring semi-supervisionate dove alcune delle istanze etichettate dall'utente possono fungere da training set per l'algoritmo.

Richiami teorici

Il seguente lavoro di tesi si pone come obbiettivo quello di introdurre un nuovo approccio al task del Clustering, introducendo la statistica come ausilio per l'individuazione di agglomerazioni anomale di dati (agglomerazioni sia positive che negative) rispetto ad una distribuzione casuale di punti .
D'altra parte se si ricerca la definizione di clustering stesso si ottiene [26] :

Il Clustering o analisi dei gruppi è un insieme di tecniche di analisi multivariate di dati volte alla selezione e raggruppamento di elementi omogenei in un insieme di dati

Quindi il clustering cerca relazioni tra i dati che portano a raggruppamenti .
La condizione 0 , ovvero l'assenza di relazioni tra i dati , determina che ogni punto sia indipendente dall'altro , caso rappresentante una distribuzione casuale di punti in uno spazio .

Casualità dei punti e Distribuzione Binomiale

Prima di parlare di distribuzione binomiale occorre una premessa :
Che cos'è un numero casuale ?

Si consideri di lanciare un dado. Fermando la situazione un attimo prima che il dado cada e mostri la faccia superiore. Finché è in aria esso costituisce un numero causale, cioè un numero che può assumere certi valori con certe probabilità.
I valori che può assumere sono 1, 2, 3, 4, 5, 6. Se non si hanno motivi per ritenere che tale dado sia truccato, le corrispondenti probabilità sono tutte uguali. E poiché la somma di tutte le probabilità deve dare 1, ciascuna di esse è uguale a 1/6.
Possiamo sintetizzare un numero casuale mediante una tabella la cui

prima riga descrive i valori che esso può assumere e la seconda riga descrive le corrispondenti probabilità.

$$\text{dado} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

Un numero casuale si chiama anche variabile aleatoria: "variabile" nel senso che può assumere diversi valori; "aleatoria" nel senso che ad ogni valore è assegnata una probabilità ("alea" sta per "dadi" in latino).
Nel caso del dado regolare, le probabilità sono tutte uguali; si dice che il corrispondente numero casuale ha una distribuzione uniforme.

Se ora si decide di ripetere la prova del lancio di dadi per 100 volte, contanto le volte in cui esce il numero 6, otterremo, in media :

100* 1/6 = 100/6 = 16,6667.

Questo numero si chiama valore atteso della variabile aleatoria.
Quindi il valore atteso del numero di volte in cui esce il numero 6 è compreso tra 16 e 17 .

In generale il valore atteso di un numero casuale

$$\begin{bmatrix} x_{[1]} & x_{[2]} & \cdots & x_{[n]} \\ p_{[1]} & p_{[2]} & \cdots & p_{[n]} \end{bmatrix}$$

è

$$x_{[1]} \cdot p_{[1]} + x_{[2]} \cdot p_{[2]} + \cdots + x_{[n]} \cdot p_{[n]}$$

oppure, utilizzando il simbolo di sommatoria:

$$\sum_{k=1}^{n} \left(x_{[k]} \cdot p_{[k]} \right)$$

L'ultimo problema risolto è un trampolino di lancio per l'approccio ad una distribuzione di probabilità molto importante, la distribuzione binomiale.
Si supponga di lanciare una moneta, le quale presenti T con probabilità t=0.7 e C con probabilità c=1−t=0.3. Lanciamola due volte. Gli eventi possibili sono :

- Al primo lancio T e al secondo C
- Al primo lanco C e al secondo T
- In tutti e due i lanci T
- In tutti e due i lanci C

Ogni volta che lanciamo la moneta, T e C hanno sempre la stessa probabilità di presentarsi (rispettivamente 0.7 e 0.3), indipendentemente da che cosa sia successo nei lanci precedenti; i lanci sono indipendenti e l'uscita di T al primo lancio non influenza il verificarsi di T anche al secondo lancio. Per questo motivo
la probabilità che, per esempio, esca T sia al primo che al secondo lancio è uguale al prodotto delle probabilità: t*t = 0.7^2 = 0.49 = 49%. Possiamo immaginare che, se al primo lancio esce T il 70% delle volte, uscirà T anche al
secondo lancio il 70% del 70% delle volte, cioè il 49%. Analogamente la probabilità che esca prima T e poi C è tc = 0.7*0.3 = 0.21 = 21%.

C'è una curiosa analogia tra quello appena descritto ed il prodotto del binomio (t+c) per sé stesso:

$$(t+c)(t+c) = t2 + tc + ct + c2 = t2 + 2tc + c2$$

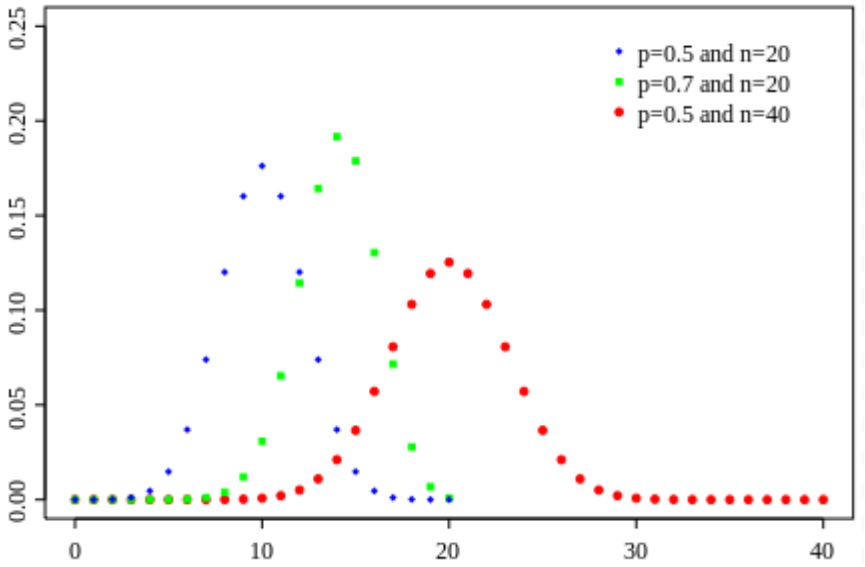
In effetti lanciare due volte una moneta è in relazione con l'elevare al quadrato un binomio: in entrambi i casi ci sono quattro possibili esiti. Quando si moltiplica un binomio per un altro binomio occorre calcolare 4 prodotti: è come se ogni volta sia necessario scegliere

una lettera nel primo binomio e una lettera nel secondo. Ci sono quattro diverse possibilità: t2, tc, ct, c2.
Non considerando l'ordine dei lanci due di queste quattro possibilità (tc e ct) sono uguali.
Se tale l'esperimento viene ripetuto *n* volte allora la probabilità di avere *k* successi

$$\mathrm{ncr}(n,k)\cdot p^{k}\cdot(1-p)^{(n-k)}$$

con ncr = N! / (N-K)!

Ora , impostando N → 20,40 e P-> 0.5 , 0.7 ed esplorando ogni singolo valore discreto si ottiene il seguente grafico



La distribuzione binomiale è quindi è caratterizzata da due parametri:

•p: la probabilità di successo della singola prova (detta prova di beroulli) *Xi* ($0 < p < 1$).
•N: il numero di prove effettuate.

Distribuzione Ipergeometrica e distribuzione ipergeometrica Multivariata.

Si supponga ora di avere a disposizione n^1 oggetti di tipo 1 e n^2 oggetti di tipo 2 (ad esempio pensiamo a palline di due colori all'interno di un urna) . Nel caso si estragga un elemento alla volta, ed ad ogni estrazione l'oggetto estratto venga re-inserito all'interno dell'urna , la variabile aleatoria X che conta il numero di oggetti di tipo 1 estratti segue una distribuzione binomiale perchè le prove sono indipendenti tra di loro.

Ma se ora si ripete la stessa prova ma senza re-inserire la pallina nell'urna ?

In questo caso si ha che X segue una distribuzione differente , in

quanto ogni evento è collegato al precedente , in quanto il numero di palline disponibili varia ad ogni prova ed anche le rispettive probabilità di successo :

$$P(X=k)=\frac{\binom{n_1}{k}\binom{n_2}{n-k}}{\binom{n_1+n_2}{n}}, \qquad k=0,1,\ldots,n$$

Tale distribuzione di probabilità è denominata Distribuzione Ipergeometrica .

Se , invece di avvere le palline di solo 2 tipi , le avessi di n tipi differenti , la distribuzione muterebbe ancora la sua forma , e verrebbe denominata “Distribuzione ipergeometrica multivariata” :

$$P(k_1,...,k_s)=\frac{\binom{h_1}{k_1}\cdots\binom{h_s}{k_s}}{\binom{n}{r}}.$$

Test di verifica di ipotesi applicati alle distribuzioni.[27]

Il test di verifica d'ipotesi è utilizzato per verificare la bontà di un'ipotesi. Per ipotesi è da intendersi un'affermazione che ha come oggetto accadimenti nel mondo reale, che si presta ad essere confermata o smentita dai dati osservati sperimentalmente.

Il metodo con cui si valuta l'attendibilità di un'ipotesi è il metodo sperimentale. Quest'ultimo consiste nel determinare le conseguenze di un'ipotesi in termini di eventi osservabili, e di valutare se la realtà effettivamente osservata si accorda o meno con l'ipotesi su di essa

fatta.

A tal riguardo si distinguono due ambiti in cui tale attività si esplica:

- deterministico;
- statistico.

Nell'ambito statistico, a seconda delle ipotesi si distingue tra:

- Test parametrico
- Test non parametrico

Nel secondo caso la situazione è modificata in quanto interviene un elemento nuovo, ovvero il caso.

Si supponga di avere una moneta recante due facce contrassegnate con testa e croce. Volendo verificare l'ipotesi di bilanciamento della moneta si eseguono 20 lanci e si contano quelli che danno esito testa. La conseguenza del bilanciamento consiste nell'osservare un valore di teste attorno a 10. Tuttavia anche in ipotesi di bilanciamento non si può escludere di osservare 20 teste. D'altronde, l'ipotesi di bilanciamento è logicamente compatibile con un numero di teste variante da 0 a 20. In tale contesto una qualsiasi decisione in merito all'ipotesi da verificare comporta un rischio di errore. Ad esempio rigettare l'ipotesi di bilanciamento della moneta avendo osservato 20 teste su 20 lanci comporta il rischio di prendere una decisione errata. Nel procedere alla verifica dell'ipotesi di bilanciamento della moneta, si ricorre a una variabile aleatoria X. Tale variabile aleatoria X rappresenta la distribuzione binomiale B(20; 0,5).

Il risultato sperimentale si deve quindi confrontare con tale distribuzione: quanto è distante tale risultato dal valore atteso della distribuzione B(20; 0,5)? Per valutare la distanza tra il valore sperimentale e quello atteso si valuta la probabilità di ottenere un valore sperimentale lontano dal valore medio di B(20; 0,5), ossìa nel caso che dal nostro esperimento risulti X=15 (15 teste dopo 20 lanci), si calcola P{|X-10|>=15-10} quindi P{X<=5 oppure X>=15}=0,041.

Quindi, usando una moneta ben bilanciata, la probabilità di ottenere un numero di teste X >= 15 (oppure X <= 5) dopo 20 lanci è pari a 0,041 ossia al 4,1%. Giudicando bassa tale probabilità si rifiuterà l'ipotesi di bilanciamento della moneta in esame, accettando quindi il rischio del 4,1% di compiere un errore nel rifiutarla. Di solito, il valore della probabilità adottato per rifiutare l'ipotesi nulla è < 0,05. Tale valore è detto livello di significatività (o confidenza) ed è definibile come segue:

Il livello di significatività sotto l'ipotesi nulla è la probabilità di cadere nella zona di rifiuto quando l'ipotesi nulla è vera.

Tale livello di significatività si indica convenzionalmente con α.

Il livello di confidenza osservato α del test per il quale si rifiuterebbe l'ipotesi nulla è detto *p-value*.

L'ipotesi da verificare si chiama ipotesi nulla e si indica con *H0*, mentre l'ipotesi alternativa con *H1*. Nel caso della moneta, se *p* è la probabilità di ottenere testa in un lancio la verifica di ipotesi si traduce nel seguente sistema:

$$H_0 : p = \frac{1}{2}$$
$$H_1 : p \neq \frac{1}{2}$$

Come già osservato, il modo di condurre un test statistico comporta un rischio di errore. Nella pratica statistica si individuano due tipi di errori:

- Rifiutare H0 quando è vera (errore di prima specie);
- Accettare H0 quando è falsa,(errore di seconda specie)

Test Chi-Quadro [28]

Il test chi quadrato si è uno dei test di verifica di ipotesi usati in statistica che utilizzano la variable casuale “Chi quadro” per verificare se l'ipotesi nulla è probabilisticamente compatibile con i

dati. A seconda delle ipotesi di partenza usate per costruire il test, tali test vengono considerati a volte parametrici e altre volte non parametrici.

I risultati ottenuti nei campioni non sempre concordano esattamente con i risultati teorici attesi secondo le regole di probabilità, anzi, è ben raro che questo si verifichi. Per intenderci, benché considerazioni teoriche ci portino ad attenderci 50 teste e 50 croci da 100 lanci di una moneta, è raro che questi risultati siano ottenuti esattamente, ma nonostante questo non si deve per forza dedurre che la moneta sia truccata. Supponendo che in un particolare campione si sia osservato che un insieme di possibili eventi E1, E2, …, Ek si presenta con frequenze o1, o2, …, ok dette frequenze osservate, e che, secondo le regole della probabilità, ci si attenda che si presenti con frequenze e1, e2, …, ek dette frequenze teoriche o attese:

Evento	**E1**	**E2**	**...**	**Ek**
Frequenze osservate	o1	o2	...	ok
Frequenze attese	e1	e2	...	ek

Lo scopo del test χ^2 è quello di conoscere se le frequenze osservate differiscono significativamente dalle frequenze teoriche.

Se $\chi^2 = 0$, le frequenze osservate coincidono esattamente con quelle teoriche. Se invece $\chi^2 > 0$, esse differiscono. Più grande è il valore di χ^2, più grande è la discrepanza tra le frequenze osservate e quelle teoriche. Nella pratica le frequenze teoriche vengono calcolate sulla base di un'ipotesi H0. Se sulla base di questa ipotesi il valore calcolato di χ^2 è più grande di un certo valore critico (come 20.95 o 20.99, che sono i valori critici rispettivamente ai livelli di significatività 5 % e 1 %), dovremmo concludere che le frequenze osservate differiscono significativamente dalle frequenze attese e dovremmo rifiutare H0 al corrispondente livello di significatività. Altrimenti dovremmo accettarla, o almeno non rifiutarla. Tale procedimento è chiamato test chi-quadrato dell'ipotesi.

Bisognerebbe notare che si deve guardare con sospetto a circostanze in cui χ^2 è troppo vicino allo zero, poiché è raro che le frequenze

osservate concordino troppo bene con le frequenze teoriche. Per esaminare tali situazioni, possiamo determinare se il valore calcolato di χ^2 è minore di 20.05 o di 20.01 nel qual caso dovremmo concludere che l'accostamento è troppo buono ai livelli di significatività del 5 % e 1 % rispettivamente.

Per conoscere i valori critici di χ^2 ad un determinato livello di significatività e con gli opportuni gradi di libertà ci si può avvalere di tabelle, oppure si possono calcolare numericamente partendo dalla corrispondente istanza della distribuzione χ^2 e calcolandone l'integrale nell'opportuno intervallo che dipenderà dal livello di significatività scelto.

Test di Kolmogorov-Smirnov [29]

Il test di Kolmogorov-Smirnovè un test non parametrico che verifica la forma delle distribuzioni campionarie. È applicabile a dati per lo meno ordinali, ossia che è possibile specificare un ordine di maggioranza / minoranza tra i dati posti in esame. Nella sua formulazione esatta prevede che le variabili siano continue. Non richiede di per sé alcuna ipotesi sulla distribuzione campionaria (salvo nel caso a un campione, in cui viene testata una distribuzione a propria scelta).

Si distingue in

- test a due code, a un campione oppure a due campioni;
- test a una coda, a un campione oppure a due campioni.

È per certi versi l'alternativa non parametrica al test t di Student[30]. Quando tale test è applicabile (ipotesi parametrica di distribuzione gaussiana) e si sceglie lo stesso il test K-S, allora l'efficienza -potenza è pari a circa il 95% per piccoli campioni e diminuisce leggermente per campioni grandi.

Rispetto ai non parametrici "test della mediana" e test del chi quadro (applicato a dati ordinali) è più potente e dunque da

preferire.

Si ritiene che per campioni molto piccoli il test K-S sia da preferire al test di Wilocoxon-Mann-Whitney[31] mentre per campioni grandi quest'ultimo è da preferire.

Sia X una variabile casuale generatrice continua, con funzione di ripartizione $F(x)$. Un problema che spesso ricorre nella pratica è quello di verificare che la variabile casuale X abbia funzione di ripartizione uguale ad una data $F_0(x)$. In simboli, il problema di ipotesi è del tipo:

$H_0 : F(x) = F_0(x), \ \forall x$

contro

$H_1 : F(x) \neq F_0(x), \ \text{per qualche } x.$

Questo significa che l'ipotesi non si riferisce soltanto ad un parametro della variabile casuale X (come accade nel test dei segni), ma all'intera sua funzione di ripartizione.

Sia allora $(X_1, ..., X_n)$ un campione casuale di ampiezza n della variabile casuale X. Sulla base di esso si vuole costruire un test per il problema di ipotesi. Poiché tale problema riguarda la funzione di ripartizione della variabile casuale X, è intuitivo basare la statistica test sulla funzione di ripartizione empirica. Dette quindi $X(1),...,X(n)$ le n variabili casuali campionarie ordinate, la funzione di ripartizione empirica è definita come:

$$\hat{F}_n(x) = \begin{cases} 0, & \text{se } x \leq X(1) \\ \frac{k}{n}, & \text{se } X(k) \leq x < X(k+1) \\ 1, & \text{se } x \geq X(n) \end{cases}$$

o equivalentemente in forma più compatta:

$$\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{X(i) \leq x}$$

dove $I_{X(i) \leq x}$ è la funzione indicatrice.

La $\hat{F}_n(x)$ è una "stima campionaria" della "vera" funzione di ripartizione $F(x)$ della variabile casuale X. Anzi, siamo in presenza di uno stimatoreconsistente, poiché si può dimostrare che, come conseguenza della legge debole dei grandi numeri, qualunque sia x

la $\hat{F}_n(x)$ tende in probabilità, per $n \longrightarrow \infty$, a $F(x)$.

L'idea del test di Kolmogorov-Smirnov è piuttosto semplice e intuitiva. Poiché $\hat{F}_n(x)$ stima la "vera" funzione di ripartizione $F(x)$, è logico basarsi su una qualche "distanza" tra $\hat{F}_n(x)$ e $F_0(x)$. Se $\hat{F}_n(x)$ e $F_0(x)$ sono "vicine" (cioè sono "sufficientemente simili") si accetta l'ipotesi nulla, mentre la si rifiuta se $\hat{F}_n(x)$ e $F_0(x)$ sono "lontane" (cioè se sono "molto dissimili"). Come "distanza" si usa la seguente:

$$D_n = \sup_{-\infty<x<+\infty} \left|\hat{F}_n(x) - F_0(x)\right|$$

cioè la massima differenza (in valore assoluto) tra la funzione di ripartizione empirica $\hat{F}_n(x)$ e la funzione di ripartizione teorica (ipotizzata come vera)$F_0(x)$. Per valori "grandi" di D_n si rifiuta l'ipotesi nulla, mentre la si accetta per valori "piccoli" di D_n (vedasi variabile casuale test di Kolmogorov-Smirnov).

Dunque, il "senso" della statistica D_n è intuitivamente evidente. Molto complicato invece è il calcolo della sua distribuzione di probabilità (sotto l'ipotesi nulla). Si può comunque dimostrare che sotto l'ipotesi nulla la distribuzione di probabilità della statistica test D_n non dipende dalla particolare forma funzionale di $F_0(x)$.

Questi risultati sono validi per le variabili casuali che hanno funzione di ripartizione continua. Se invece X è una variabile casuale discreta e la sua funzione di ripartizione è quindi discontinua, la distribuzione di probabilità della variabile casuale D_n dipende proprio dalla discontinuità della funzione di ripartizione di X.

Algoritmi di clustering basati su metodi statistici

In questa sezione saranno esaminati tutti e tre gli algoritmi sviluppati durante il lavoro di tesi , le motivazioni che hanno portato a determinate scelte e una breve discussione sui risultati ottenuti in fase implementativa .

É stato necessario sviluppare e testare più algoritmi per poter avere una maggiore comprensione della problematica di confrontare una serie di punti dislocati in uno spazio d-dimensionale con la stessa distribuzione spaziale ma con valori casuali .

Le scelte strutturali alla base di tutti gli algoritmi sono state :

- Orientamento Grid-Based dell'algoritmo , con suddivisione di ogni dimensione in N regioni
- Utilizzo dello stesso algoritmo di clustering per tutti e tre gli algoritmi
- I parametri di ingresso quali N appena descritto e la confidenza del test di Kolmogorov Smirnov.

Tali scelte architetturali possono portare a diverse problematiche , le quali saranno analizzate nel capitolo successivo, ma l'obbiettivo di tesi è stato più concentrato sugli algoritmi di individuazione di regioni Casuali o non Casuali piuttosto che sulla ricerca dell'ottimo criterio di divisione delle d dimensioni o sul clustering, ottenendo comunque buoni risultati.

Concetti di regione casuale, Densa+ , Densa -

Le regioni posso appartenere a due tipologie:

- Regioni Casuali
- Regioni Non casuali

Una regione casuale è una regione per cui , rispetto al suo intorno o all'intero dataset, risulta a meno di un margine di errore rispecchiare la rispettiva distribuzione casuale . Saranno quindi etichettate come “Casuali” tutte le regioni che , a seconda dell'algoritmo utilizzato, risulteranno compatibili con la distribuzione Casuali dei punti del dataset.

Una regione Non casuale è invece una regione che non rispetta il pattern di casualità ; occorre però considerare che ve ne sono di due tipi :

- Non Casuali Dense → il numero di punti non rispecchia il pattern ma rappresenta una regione ad elevata numerosità , che d'ora in poi saranno chiamate Dense +
- Non Casuali non Dense → il numero di punti non rispecchia il pattern ma rappresenta una regione a bassa numerosità , che d'ora in poi saranno chiamate Dense -

Orientamento Grid-Based ed adiacenza d'intorno

Come già accennato si è deciso di utilizzare un approccio Grid-Based al problema, ovvero quello di suddividere il data-set in R regioni .

Una regione è formata da :

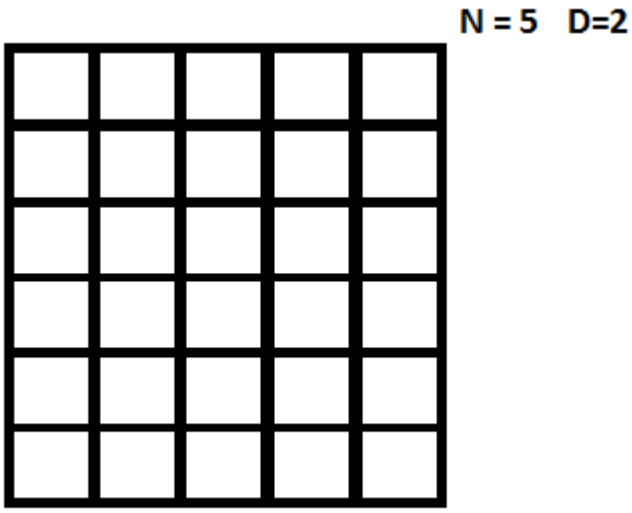
- Un identificativo
- La Numerosità ovvero il numero di punti che “cadono” in tale regione

Non si è deciso quindi di creare una struttura dati complessa per ottimizzare al massimo il consumo di memoria .

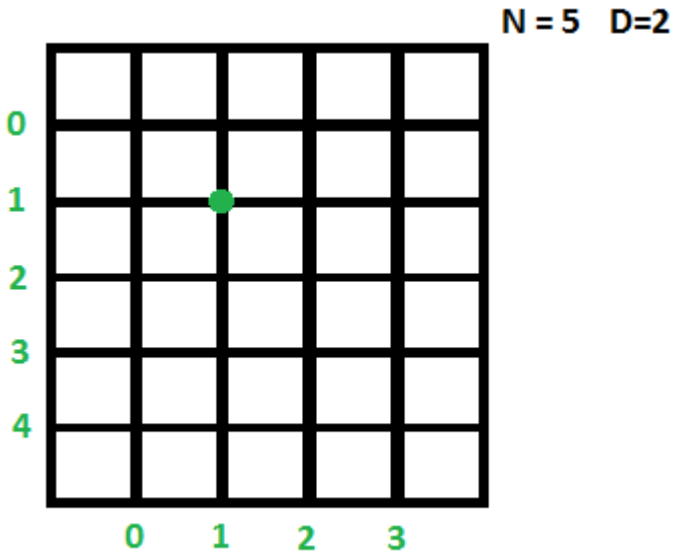
Suddivisione dei punti all'interno di una regione

Si è deciso di suddividere lo spazio in R regioni , utilizzando N quale parametro di input:
N rappresena il numero di suddivisioni per ogni dimensione , quindi con N = 20 e D = 2 (due dimensioni nel dataset) si ottengono 400 regioni .
In generale quindi :

$$R = N \wedge D$$



Osservando la figura si nota come l'area totale debba essere divisa in 5 sezioni per ognuna delle due dimensioni .



In memoria sarà mantenuto il solo "puntatore di incrocio" , ovvero l'identificatore della cella , senza persistere una struttura dati complessa quale una griglia contenente i valori Max/Min di range per ognuna delle regioni ottenendo quindi un'attribuzione istantanea .

Quindi , considerando il caso in cui D = 2 ed N =5 otteniamo che :

- La prima riga è rappresentata dal'indice di riga 0 e l'indice di colonna che varia da 0 a 4
- La seconda riga è rappresentata dall'indice di riga 1 e l'indice

di colonna che varia da 0 a 4

- Ecc per le altre righe.

In generale non si ottiene mai un indice di Riga superiore a 4 e pure un indice di colonna.
Tale configurazione rappresenta un sistema metrico su base 5 , dove le unità rappresentano le colonne e le righe rappresentano le "decine" .
Quindi per identificare la regione R(1,1) occore semplicemente eseguire la seguente moltiplicazione :

$$C(X,Y) = X * 5^0 + Y*5^1 = 6 .$$

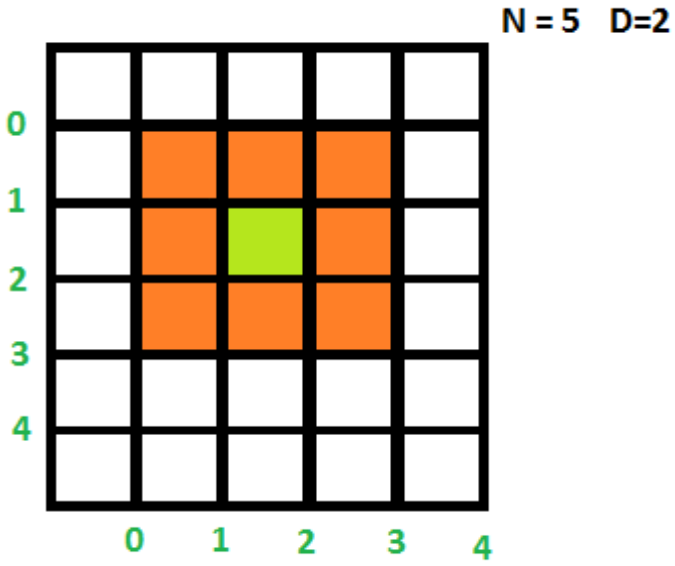
Il numero 6 trovato rappresenta l'indice della regione R all'interno del dataset.
Generalizzando , nel caso multi-dimensionale dato N parametro di input e D calcolato in base agli attributi del dataset si ottiene :

$$C(d0,d1,....,dD) = d0*N^0 + d1*N^1+d2*N^2 + ... + dD*N^D$$

Risulterà infine un unico Array monodimensionale contenente tutte le regioni.

Adiacenza d'intorno

Un'attività che potrebbe risultare banale ma che comporta un elevato ritorno sia in termini di memoria ma soprattutto a livello computazionale è la deifinizione di "intorno" di una regione ; questo perchè molte delle considerazioni che saranno esplicate nei prossimi paragrafi utilizzeranno il concetto di intorno o di "vicinato" .



Per questa prima implementazione è stato scelto di considerare intorno di una regione le proprie celle adiacenti , ovvero quelle "raggiungibili" da una qualsiasi regione R incrementado / sottraendo al massimo di 1 ogni dimensione .
Tale assegnazione comporterà quindi una selezione di un vicinato di 8 regioni per un totale , includendo la regione di partenza , di 9 regioni selezionate .
Considerando sempre il caso generale , chiamando I l'intorno di una regione R in D dimensioni si ottiene :

$$I = 3^D$$

Algoritmo 1

Il primo algoritmo implementato segue la logica di confrontare l'intero dataset osservato con il rispettivo data-set casuale considerando la sola distribuzione di numerosità all'interno delle regioni , ovvero effettua un raffronto su come si distribuisce la numerosità delle regioni .

I passaggi effettuati dall'algoritmo , in pseudo codifica , sono :

Input : dataset , N , confidenza

```
D = CalcolaNDimensioni(dataset)
Nregioni = N ^ D
Npunti = dataset.Count
Regioni = AttribuisciPuntiARegioni(dataset,Nregioni)
DistribuzioneOsservata=CalcoloDatiOsservati (dataset , Regioni)
DistribuzioneTeorica = CalcoloDatiAttesi(Npunti , Nregioni )

while(Confronta(DistrOsservata,DistrTeorica) == false)
        RegionElimina=CalcoloMaxDifferenza(DistrOsservata,DistrTeorica)
       DistrOsservata = DistrOsservata - RegionElimina
        Npunti = Npunti – punti(RegionElimina)
        Nregioni = Nregioni -1
        SettaNonCasuale+o-(Regioni[RegionElimina])
        DistribuzioneTeorica=DistribuzioneBinomiale(Npunti, Nregioni)

EseguiCluster(Regioni).
```

Preparazione del campione casuale

Prendendo in esame quanto descritto per il caso della distribuzione binomiale, è facile intuire come l'attribuzione di un punto casuale all'interno di uno spazio suddiviso in N regioni consiste in una serie di prove indipendenti (e quindi bernoulliane) che hanno come N il numero di punti del dataset e come P la possibilità che un determinato punto ricada all'interno di una regione R ovvero 1/nRegioni .

Ottenuta la distribuzione binomale è necessario calcolare i valori attesi per ogni regione , ovvero si costruisce un'array avente come indice la numerosità dei punti all'interno della regione e come valore la numerosità delle regioni di quel tipo .

Quindi un'array di Dati teorici TA [0,2,5,3,1,0] sarà rappresentativo

di :

- 0 regioni da 0 punti
- 2 regioni da 1 punto
- 5 regioni da 2 punti
- 3 regioni da 1 punto
- 0 regioni da 0 punti

L'array finale segue la distribuzione binomiale , ad esempio per Npunti = 8000 e Nregioni = 2500 si ottiene:

Ta[102, 326, 522, 557, 445, 285, 152, 69, 27, 10, 3, 1, 0]

da dati teorici infatti la media è P*N = 1/2500 * 8000 = 3,2 e si nota come la maggiore numerosità di regioni dello stesso tipo coincide con la media della binomiale.

Preparazione del campione osservato

Attribuiti i punti alle regioni come descritto in 4.2 si esegue il calcolo dei valori osservati .

L'array risultante deve essere dello stesso tipo di 4.3.1 , si calcola quindi semplicemente quanti tipi di regioni si hanno in base alla propria numerosità.

Test di conformità tra distribuzioni

Preparate entrambe le distribuzioni di numerosità esse vengono confrontate tramite il test di Kolmogorov-Smirnov descritto in 3.3.2 fissato un margine di confidenza da parmetro di input . Il risultato del test è quindi confrontato con il valore critico [32] del test , se tale test ha esito positivo le due distribuzioni “fittano” , quindi tutti i dati rimasti rappresentano una distribuzione casuale di punti.

Ricorsione

Al primo check è praticamente impossibile che la distribuzione teorica e la distribuzione osservata “fittino” , questo a meno che non si testi direttamente una distribuzione casuale di punti .

Se quindi il test da esito negativo si procede con l'eliminazione delle regioni Non casuali tramite i seguenti passaggi:

- Viene calcolata la massima differenza tra le celle dei due array con stesso indice , si indaga quindi su qual'è il componente che ha contribuito maggiormente al fallimento del test .
- Data la massima differenza si individua la tipologia di regione a cui va eliminata una regione ; quindi delle regioni ottenute si calcola, considerando le regioni tra loro indipendenti , qual'è la regione che ha la minor probabilità di essere casuale ; tale considerazione è fatta interrogando direttamente la binomiale dei dati casuali .
- Individuata la regione essa viene tolta dall'esame e marchiata come NON casuale , se essa è maggiore della media della distribuzione è marchiata come NON casuale + mentre se è inferiore è marchiata come NON casuale - .
- Una volta tolta la regione occore ricalcolare la distribuzione binomiale considerando sia la regione in meno che il numero dei punti attribuiti alla regione eliminati.
- Viene quindi rieseguito il test di KS , se esso ha ancora esito negativo viene rieseguita la procedura, altrimenti si passa alla fase di clustering.

Problematiche

Dalla prima fase di verifica si è evinto subito una forte

problematica, dovuta ad una carenza strutturale dell'algoritmo .

Difatti non si tiene conto in questo algoritmo della distribuzione spaziale dei punti all'interno del dataset, si elimina puntualmente ogni regione , lasciando quindi le sole regioni che potrebbero “fittare” con una distribuzione casuale, ma in questo modo i “buchi” non sono più considerati , e quindi incorrerei in assegnazioni errate di regioni casuali.

Per poter meglio comprendere quanto detto si faccia riferimento al seguente esempio :

Prendendo in esame una qualsiasi distribuzione casuale in un determinato spazio , ottengo una certà numerosità di celle .

Se ora però seleziono tutte le celle di una certa elevata numerosità ottenute e le metto tutte adiacenti le une alle altre, facendo anche la stessa operazione con quelle a minore numerosità , ecco che ho creato un'area non casuale + ed una non casuale - .

è quindi indispensabile considerare anche la distribuzione di spaziale delle regioni, soprattutto le relazioni con il proprio vicinato.

Algoritmo 2 : Densità e distribuzione

Per porre quindi rimedio alle carenze della versione “1.0” dell'algoritmo di clustering statistico sono state apportate diverse modifiche .

In questa versione è stata rimossa l'eliminazione ricorsiva delle regioni ma si è tentato di effettuate un match diretto tra la distribuzione casuale dei punti e la distribuzione teoria secondo due parametri:

- Probabilità di una cella di essere casuale rispetto alla densità
- Probabilità di una cella di essere casuale rispetto alla numerosità del proprio vicinato.

I primi passaggi ed i parametri di input sono rimasti immutati, è stato modificato anche il test , non eseguito più a livello generale ma

a livello di intorno di cella .

Segue la pseudo-codifica della versione 2 dell'algoritmo

```
Input  : dataset , N , confidenza
D = CalcolaNDimensioni(dataset)
Range = CalcolaRangeDimensioni(dataset)
Nregioni = N ^ D
Npunti = dataset.Count
Regioni = AttribuisciPuntiARegioni(dataset,Nregioni)
DatasetCasuale = CalcolaDataSetCasuale(Npunti,Range)
RegioniCasuali = AttribuisciPuntiARegioni(DatasetCasuale,Nregioni)
DistrVarianze = CalcolaDistribuzioneCasualeVarianze
                                    (datasetCasuale,Npunti)
Foreach regioneCasuale di RegioniCasuali
        CalcolaVarianzaIntorno(regioneCasuale)
        CalcolaProbabilitàVarianzaCasuale(regioneCasuale)
Foreac regione di Regioni
        CalcolaProbabilitàSuDistribuzioneDensità
        CalcolaVarianzaIntorno(regione)
        CalcolaProbabilitàVarianzaCasuale(regione)
        ProbabilitàTotale =regione.ProbVarianza * regione.ProbailitàDensa
```

ValoreAttesoTotale = ProbabilitàTotale Nregioni^2.*

Foreach regione di Regioni

CalcolaMarchiaturaSuValoriAttesiTotali
(regione,regioneCasualeCorrispondente).

EseguiCluster(Regioni).

Preparazione del campione casuale

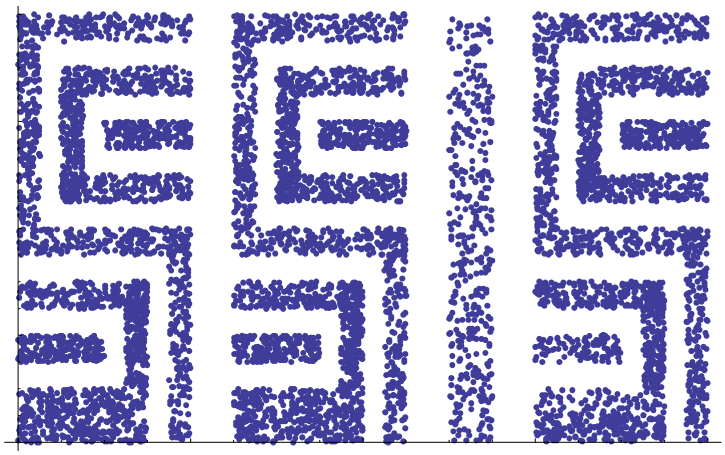
La preparazione del campione casuale differisce da quanto eseguito nel primo algoritmo in quanto ogni volta, in base al Npunti ed allo spazio del dataset (range di ogni dimensione), viene ricreata una distribuzione casuale di punti attraverso l'utilizzo dei componenti Random (che sono presenti in ogni libreria Java, C# , ecc) .

Si crea quindi un nuovo dataset che servirà soprattutto a creare la distribuzione delle variazioni di numerosità delle celle .

Tale nuovo dataset rappresenta il “datasetTeorico” di confronto sul quale poi saranno raffrontate le arree di ogni regione.

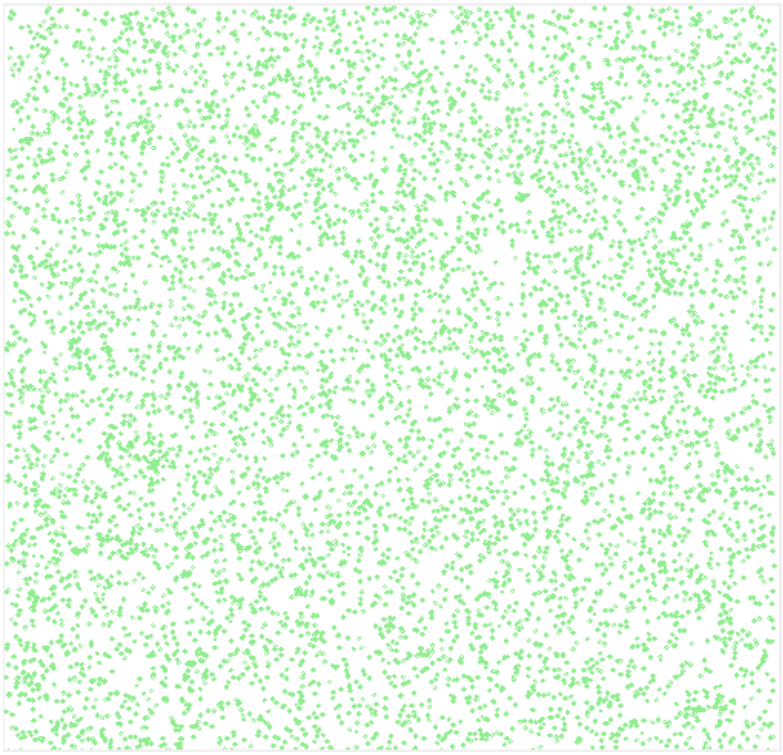
Un esempio esplicativo grafico è il seguente :

Sia dato il seguente dataset



L'occhio umano è molto abile a riconoscere i cluster fino alle 3 dimensioni.

L'algoritmo, preso in ingresso tale dataset , lo confronta con il suo rispettivo datasetCasuale .



Calcolo della distribuzione di densità

La distribuzione di densità è ottenuta seguendo la distribuzione binomiale come già visto nell'algoritmo 1.

Calcolo della distribuzione di variazione

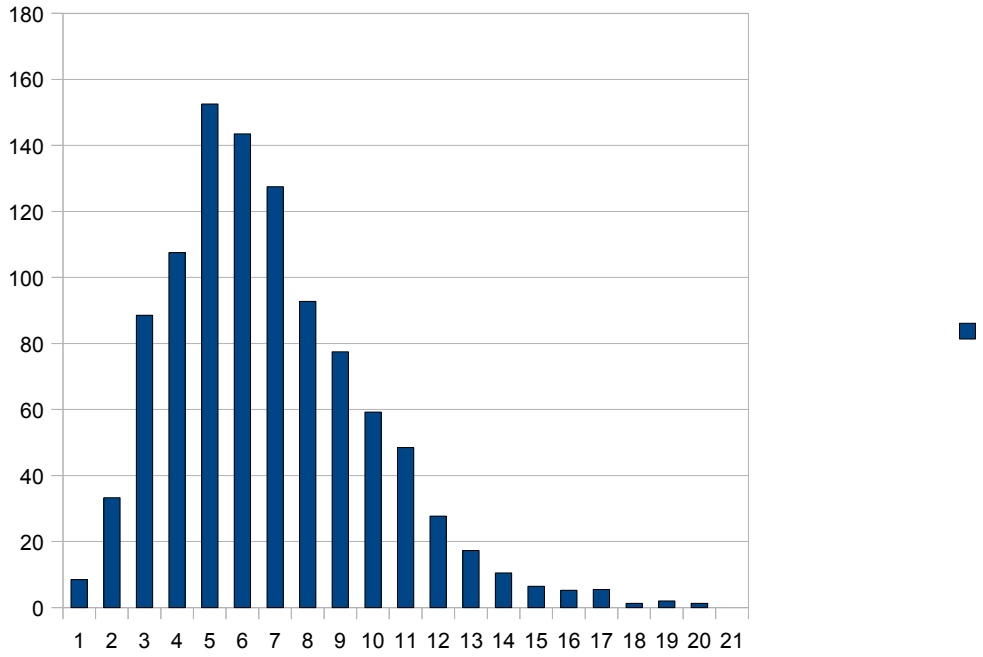
Per il calcolo della distribuzione della variazione l'algoritmo segue i seguenti passi :

- Posizionamento su una regione del dataset .
- Da essa prelevo il campione di celle adiacenti come spiegato in 4.2 creando quindi un mini-dataset formato da N^D dimensioni.
- Calcolo la densità media del mini-dataset.
- Calcolo la varianza di ogni regione del mini-dataset , dato Xi = numerosità della regione e U la media del mini-dataset

$$\sigma_X^2 = \frac{\sum_i (x_i - \mu_X)^2}{n},$$

- Dati i valori puntuali di varianza per ogni regione è possibile quindi generare la distribuzione delle varianze “Osservate” del dataset Casuale, utilizzando la stessa funzione per il calcolo della distribuzione della numerosità osservata ma applicata alla proprietà Varianza anziché Numerosità.

NOTA BENE



A seguito di prove effettuate si è notato che , come era possibile aspettarsi , la distribuzione della varianza è anch'essa a forma di distribuzione binomiale .

La figura rappresenta la media di 10 calcoli di varianza applicati ad una distribuzione casuale di 7600 punti su 2500 regioni .

Si nota come il valore complessivo della varianza ottenuta secondo il calcolo sopra descritto sia compreso tra 1 e 21 , valori decisamente bassi che indicano come all'interno di una distribuzione casuale di punti essi tendono ad essere uniformi spazialmente.

Preparazione del campione atteso

Ottenuto il campione casuale si ricavano le stesse informazioni (varianza e numerosità) per di ogni regione del campione atteso attenendosi alle modalità descritte in 4.3. per la numerosità ed in 4.4.1.1. per la varianza ,ovviamente applicate al dataset Reale.

Si ricavano due probabilità confrontando poi i dati osservati con le distribuzioni di varianza e numerosità .

- Probabilità rispetto alla numerosità , che viene calcolata interrogando la distribuzione casuale delle numerosità , ovvero

la probabilità che una regione contente un certo numero di punti possa esistere all'interno di un dataset casuale.

- Probabilità rispetto alla varianza, ottenuta interrogando la distribuzione casuale della varianza, ovvero la probabilità che una regione avente una certa varianza possa esistere all'interno di un dataset casuale.

Tali 2 probabilità sono poi moltiplicate tra loro, in quanto una regione per poter essere determinata Casuale o non deve rispettare sia i canoni di numerosità che i canoni di varianza rispetto alle altre regioni.

Si ottiene quindi una probabilità totale rappresentativa della casualità della regione singola , il cui valore atteso è ottenuto moltiplicando tale probabilità per il quadrato del numero di regioni.

Tale valore atteso è utile a titolo statistico per la verifica successiva.

Individuazione regioni Dense +/-

La casualità o meno di una cella è quindi , in base a questo algoritmo, decisa non da 1 ma da 2 fattori , numerosità e varianza.

Un ulteriore termine di paragone è l'intorno , ovvero selezionato uno spazio più grande della regione in esame è utile confrontare le probabilità “totali” di tale intorno con quelle ottenute dai dati casuali .

Viene quindi effettuato il KS test sulle regioni dell'intorno casuale

quali dati attesi e l'intorno delle regioni osservate quali dati osservati; posto il margine di confidenza quale parametro di input.

L'attribuzione delle regioni dense + o dense – una volta ottenuta la divisone tra Casuali e Non casuali avviene come descritto in 4.3 .

Algoritmo 3: Calcolo della probabilità con dipendenza

La versione “3.0” dell'algoritmo si pone un obbiettivo più ambizioso, ovvero quello di sfruttare meno il dataset dei punti Casuali senza dover ripetere il “giro” in due parti (prima vengono ciclate le regioni casuali e poi quelle non) ma inserendo le informazioni di numerosità e varianza rispetto all'intorno della regione osservata in un unico passaggio .

Inoltre cerca di colmare un piccolo “buco” della versione 2.0 ovvero quello che tale versione tiene in considerazione nei propri calcoli il solo intorno di ogni regione , escludendo di fatto il resto del dataset.

Questa valutazione potrebbe portare ad una peggiore qualità del cluster qual'ora un dato che “localmente” è possibile essere casuale non lo sia più se inserito in un contesto maggiore.

Quindi il terzo algoritmo cerca di prendere le buoni considerazioni fatte sia per la prima versione in cui si tiene conto dell'intero dataset che della seconda in cui si tiene conto di numerosità e varianza degli intorni delle regioni.

Tale algoritmo pone le sue basi nella distribuzione ipergeometrica , ovvero si effettua (così come effettuato nell'algoritmo 1.0) un'indagine senza re-introduzione delle regioni del dataset. Questo perchè la casualità di una regione è dovuta a come le altre regioni sono distribuite nello spazio, e considerare ogni intorno “indipendente” come avviene nella versione 2.0 non è corretto.

La pseudo-codifica della terza versione dell'algoritmo è la seguente :

Input : dataset , N , confidenza

```
D = CalcolaNDimensioni(dataset)
Range = CalcolaRangeDimensioni(dataset)
Nregioni = N ^ D
Npunti = dataset.Count
Regioni = AttribuisciPuntiARegioni(dataset,Nregioni)
DatasetCasuale = CalcolaDataSetCasuale(Npunti,Range)
RegioniCasuali = AttribuisciPuntiARegioni(DatasetCasuale,Nregioni)
DistrVarianze = CalcolaDistribuzioneCasualeVarianze
                                    (datasetCasuale,Npunti)
Foreac regione di Regioni
        CalcolaDensitaMedia(regione)
OrdinaRegioniSecondoDensitaENumerosita(regioni)
While (regioni ==0)
        regione = PrendiPrimaRegione
        CalcolaProbabilitàRegione(regione)
        Npunti = Npunti – punti(RegionElimina)
        Nregioni = Nregioni -1
IntornoRegioniCasuali = CalcolaIntornoCasuale(regioniCasuali)
Foreach regione di Regioni
        CalcolaMarchiaturaSuValoriAttesiTotali
                (regione,IntornoRegioniCasuali).
```

Preparazione del campione casuale

La preparazione del campione casuale rispecchia quanto descritto in 4.4.1 , viene anche in questo caso creato il rispettivo dataset di dati casuali dal quale sono poi calcolate le regioni casuali .

Di queste regioni viene poi preso in esame un campione di 10 intorni al quale sarà applicato l'algoritmo di stima della probabilità che sarà definito in 4.5.3 ; l'output di questo metodo è un' array rappresentante le probabilità di casualità attese per il test di Kolmogorov Smirnov in fase di valutazione delle regioni casuali / non casuali.

Preparazione dello stack di regioni

Per il discorso di dipendenza delle regioni dall'intero dataset , si è deciso che le regioni più dense sono quelle al quale sottoporre l'indagine di probabilità per prime, anche perchè sono quelle che interessano di più al fine del clustering e sulle quali è necessario prestare la dovuta attenzione ; essendo l'algoritmo un'approssimazione della distribuzione ipergeometrica che altrimenti richiederebbe costi computazionali elevatissimi , è quindi doveroso impostare una priorità di investigazione sulle regioni.

Le regioni sono quindi ordinate secondo densità e numerosità , con l'attributo di densità dominante sulla numerosità , ovvero a parità di densità viene selezionata la regione con maggiore numerosità.

Algoritmo di assegnazione probabilità dipendente ricorsivo.

L'algoritmo procede una regione alla volta .

Viene presa la regione in cima allo stack, e di essa viene considerato il proprio intorno.

La probabilità che tale regione sia casuale viene determinato come il produttorio delle probabilità di casualità di ogni regione rispetto alla

distribuzione di numerosità casuale.

Quindi , prendendo come esempio un intorno del tipo :

{3,1,3,2,2,4,5,6,6} ovvero un intorno di un campione bi-dimensionale il calcolo da effettuare è il seguente :

P[3][P1]*P[3]*P[2]*P[2]*P[4]*P[5]*P[6]*P[6]* .

Tale probabilità assume l'indipendenza nell intorno, e dato che la distribuzione ipergeometrica per una sola estrazione coincide con la ditribuzione binomiale si può parlare del risultato ottenuto quale APPROSSIMAZIONE della distribuzione ipergeometrica multivariata all'intorno di regione.

La probabilità risultante rappresenta quindi la probabilità che un intorno formato da :

- 2 regioni a numerosità 3
- 1 regione a numerosità 1
- 2 regioni a numerosità 2
- 1 regione a numerosità 4
- 1 regione a numerosità 5
- 2 regioni a numerosità 6

sia casuale .

Tale calcolo tiene conto quindi SIA delle numerosità delle varie regioni CHE delle veriazioni intra-regioni.

Ottenuto il risultato viene salvato sia il valore di probabilità P[2] del centroide dell'intorno che il valore totale di probabilità .

La regione è successivamente eliminata dallo stack, viene decrementato il contatore delle regioni e ricalcolata la distribuzione di probabilità binomiale in base ai punti rimasti all'interno dei punti rimasti.

Gestione dei buchi

L'algoritmo 3.0 si può definire un algortimo con “memoria” , perchè una regione viene eliminata dallo stack delle regioni da calcolare ma permane all'ìnterno dell'array delle regioni .

Questo perchè prima o poi si andrà a calcolare la probabilità anche per una delle regioni adiacenti ad una regione appena calcolata.

Quando si incontra una regione già calcolata non si esegue più l'interrogazione alla binomiale , ma si tiene conto della probabilità di casualità della cella al momento della propria prima interrogazione .

Riprendendo l'esempio precedente

{3,1,3,2,[2],4,5,6,6} =
*P[3]*P[1]*P[3]*P[2]*P[2]*P[4]*P[5]*P[6]*P[6] .*

Si ponga X[2] = P[2] .

Ora , considerando uno step successivo l'interrogazione dell'intorno.

{2,[2],4,2,{5},3,2,4,4}.

Tracciando tra “ [] “ la regione già calcolata e tra “ {} “ il nuovo centroide, il calcolo della probabilità sarà il seguente :

PT {2,[2],4,2,{5},3,2,4,4} =

*P[2]*X[2]*P[4]*P[2]*P[5]*P[3]*P[2]*P[4]*P[4] .*

Utilizzando quindi X[2] al posto di P[2] , analogamente all'esempio delle palline in un urna sarebbe come calcolare una pallina già estratta, quindi si prende in considerazione la probabilità della pallina di essere estratta al momento della propria estrazione.

Individuazione regioni NON CASUALI +/-

La probabilità ottenuta , essendo un prodotto di N probabilità , rappresenta un valore estremante piccolo , valore per il quale

sarebbe impossibile effettuare il test di KS .

Quindi sia i valori attesi che i valori osservati sono moltiplicati per un fattore F comune , calcolato in base al valore più piccolo ottenuto dal campione casuale.

Se quindi il campione di intorno casuale (calcolato come media di 10 campioni di intorni di regioni casuali) ha , ad esempio, un valore uguale $1{,}23*10^{-6}$, prima del calcolo del test di KS ogni valore casuale ed osservato è moltiplicato per 10^6.

Come effettuato nella versione 2.0 il test è effettuato tra le regioni dell'intorno casuale quali dati attesi e l'intorno delle regioni osservate quali dati osservati; posto il margine di confidenza quale parametro di input.

Clustering

L'algoritmo di Clustering, avendo a disposizione per ogni regione l'informazione di Casualità , Densa + o Densa – risulta un semplice ricercatore di percorso :

- Si pone il numero di cluster C =1
- Data una regione Densa +, essa viene marchiata apparentente al cluster C
- Si controlla il suo intorno e si marchiano a C tutte le regioni dense +
- Se una regione è Densa – ma il suo intorno ha più del 70% di regioni dense + essa rappresenta un “buco” del cluster e quindi viene attribuita al Cluster C.
- Si sposta il controllo alla regione successiva densa + collegata a C non considerando nello spostamento le eventuali dense-aggiunte al cluster (una di quelle che sono state appena attribuite a C) e si riesegue dal punto 2
- Se non vi sono altre regioni non marchiate collegate si incrementa C e si torna al punto 2 ciclando le regioni del dataset .

Test , confronti e valutazioni.

Per i motivi descritti al punto 4 gli algoritmi portati alla fase di test sono stati le sole versioni 2 e 3 , in quanto la prima versione , per la propria mancanza di elementi , non contribuiva ad una corretta valutazione dei cluster in esame .
I dataset in esame sono tutti dataset Bi-dimensionali , in quanto l'algoritmo studiato si pone come base per uno studio di clustering statistico , ed è privo di ottimizzazioni sul campo della multi-dimensionalità .
L'obbiettivo è quello di dimostrare che l'utilizzo di metodi statistici applicati al clustering può essere un'arma vincente , e tale algoritmo potrebbe essere inoltre incluso in altri rami del Grid-clustering quale il subspace-clustering ; tali argormentazioni saranno discusse successivamente.
Per i motivi appena descritti i test saranno di tipo qualitativo , ovvero non saranno considerate tempistiche e consumo di memoria ma solo la qualità dei cluster risultanti tramite visualizzazione grafica e l'utilizzo dell'indice di Rand.

Algoritmi di Confronto .

Per il confronto sono stati utilizzati alcuni degli algoritmi più comuni in letteratura quali :

- K-Means[33] →L'algoritmo **K-Means** è un algoritmo di clustering partizionale che permette di suddividere un insieme di oggetti in K gruppi sulla base dei loro attributi. È una variante dell' Algoritmo di aspettazione-massimizzazione (EM) il cui obiettivo è determinare i K gruppi di dati generati da distribuzioni gaussiane. Si assume che gli attributi degli oggetti possano essere rappresentati come vettori, e che quindi

formino uno spazio vettoriale. L'obiettivo che l'algoritmo si prepone è di minimizzare la varianza totale intra-cluster. Ogni cluster viene identificato mediante un centroide o punto medio. L'algoritmo segue una procedura iterativa. Inizialmente crea K partizioni e assegna ad ogni partizione i punti d'ingresso o casualmente o usando alcune informazioni euristiche. Quindi calcola il centroide di ogni gruppo. Costruisce quindi una nuova partizione associando ogni punto d'ingresso al cluster il cui centroide è più vicino ad esso. Quindi vengono ricalcolati i centroidi per i nuovi cluster e così via, finché l'algoritmo non converge.

- DBSCAN [20] è un algoritmo di clustering basato su densità, in particolare fa parte dell'insieme degli algoritmi basati su connettività ovvero che computano la densita sui singoli oggetti del dataset. Vedi 2.1 per maggiori dettagli.

- **COBWEB[34] è un sistema incrementale per il clustering gerarchico.** Cobweb incrementalmente organizza le osservazioni in un albero delle classificazioni , ogni nodo in una classficazione rappresente un conetto ed è marchiato come un probabile concetto che somma le distribuzioni dei valori delle distribuziooni degli oggetti classificati sotto quel nodo.

  Vi sono 4 operazioni base che COBWEB implementa nella costruzione dell'albero;

  - Merge di 2 nodi
  - Split di un nodo
  - Inserimento di un nuovo nodo
  - Passaggio di un oggetto nella gerarchia.

Indice di Rand [35]

L'indice di Rand o Rand Index o Misura di Rand nel clustering di dati è una misura di similarità tra due clustering. Dal punto di vista matematico l'indice di Rand è relativo all'accuratezza del clustering,

ma è comunque applicabile quando le “classi etichette” non sono utilizzate.
Dato un set di n elementi $S = \{o_1, \ldots, o_n\}$ e due partizioni di S da confrontare$X = \{X_1, \ldots, X_r\}$, una partizione di *S in r sottosezioni*, e $Y = \{Y_1, \ldots, Y_s\}$, una partizione di *S* in *s sottosezioni*, si definisce che :

•a,il numero delle coppie di elementi in Sche sono uguali sia nel set X che nel set Y

•b,il numero delle coppie di elementi in S che sono diverse tra loro sia nel set X che nel set Y

•c, il numero delle coppie di elementi in S che seno uguali in X ma diversi in Y

•d,il numero delle coppie di elementi in S che sono diverse in X ma uguali in Y

L'indice di rand è calcolato come segue:

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}$$

Intuitivamente a+b può essere considerao come il numero di elementi concordi tra X e Y e C+D il numero di elementi discordi.

Tale indice è utilizzato per confrontare i dataset ottenuti dagli algoritmi utilizzati nelle fase di test con un algoritmo di clustering ideale , definito dall'utente , il quale “etichetta” punto per punto a quale clustering appartiene .

Quello che verrà valutato è il numero di istanze non correttamente attribuite , ovvero il numero di errori che l'algoritmo ha effettuato rispetto al caso ideale.

Test DatasetS_ConRumore

Caratteristiche del Dataset :

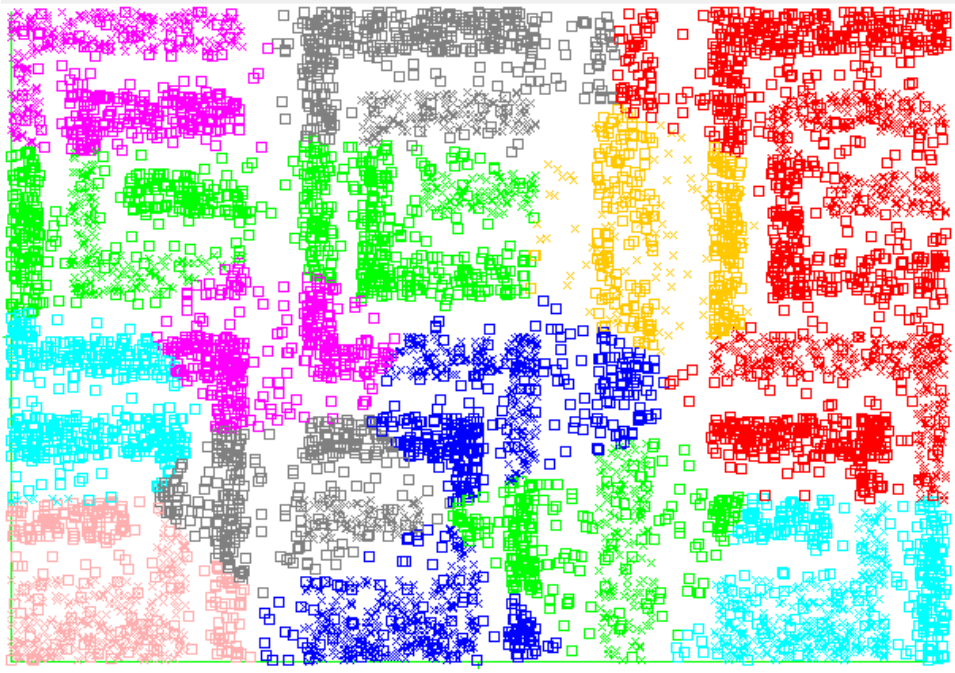
- Numero di dimensioni → 2
- Numero di istanze → 7600
- Numero di cluster → 16 + rumore

Risultati :

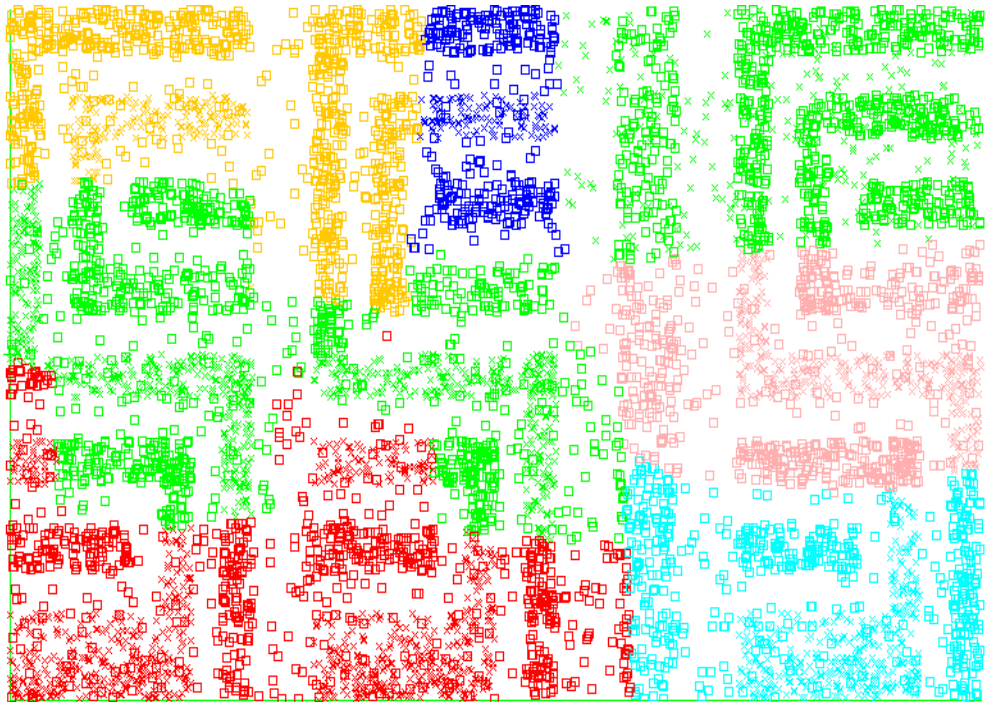
- Clustering statistico versione 2 → N=32 Confidenza =0.7
  - Numero di istanze errate → 997 ovvero il 13,013%
- Clustering statistico versione 3 → N=32 Confidenza =0.10
  - Numero di istanze errate → 1000 ovvero il 13,16%
- DBSCAN E 0.017 MinPts 6
  - Numero di istanze errate → 2180 ovvero il 28,69%
- CobWeb → A 0.90 C 0.0038
  - Numero di istanze errate → 4799 ovvero il 67,19%
- Kmeans → Seed =100 NumCluster = 16
  - Numero di istanze errate → 4579 ovvero il 60,26%

Risultati Grafici Degli algoritmi

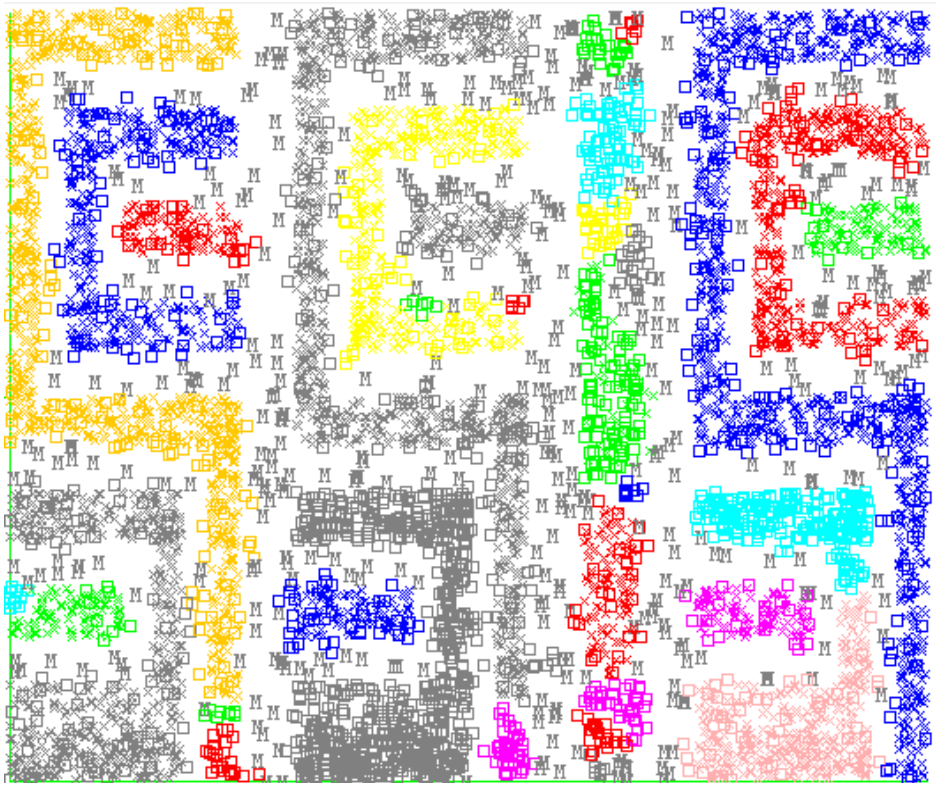
K-means



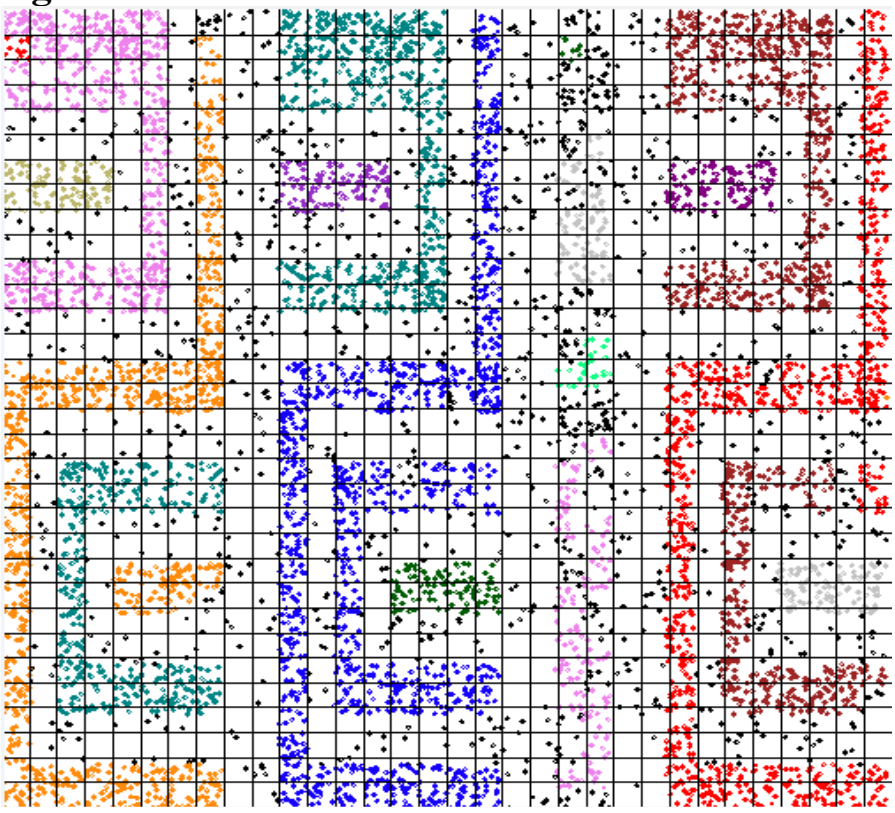
CobWeb



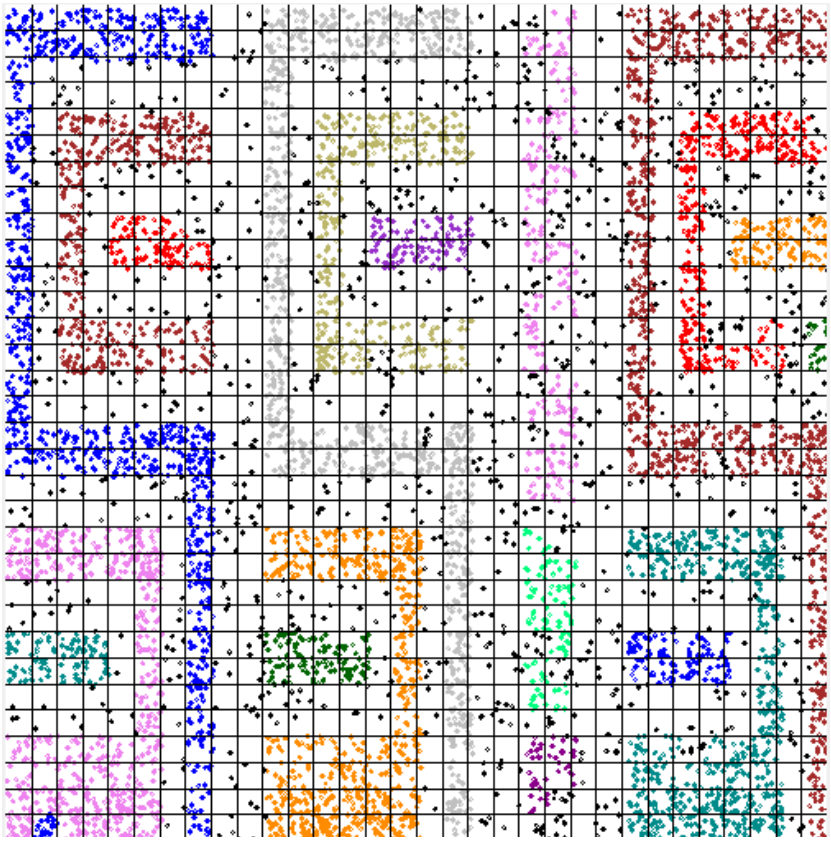
DB scan



ClusteringStatistico V 2.0



Clustering statistico 3.0



Considerazioni

Dai primi risultati si evidenziano subito i primi problemi legati alla separazione dei cluster ; difatti la semplice separazione Maggiore o minore della media in alcuni casi non basta come per il cluster centrale delle fig. sopra.

Un altra problematica è legata all'ottimizzazione della separazione delle regioni , difatti in base al numero di regioni i risultati possono essere nettamente differenti, tale numero di regioni è però individuabile secondo qualche criterio di ricerca operativa, non indagato in questo lavoro.

L'aspetto positivo delle prove effettuate invece è la resistenza al rumore e un 'ottimo riconoscimento di cluster anche di forme non solo convesse .

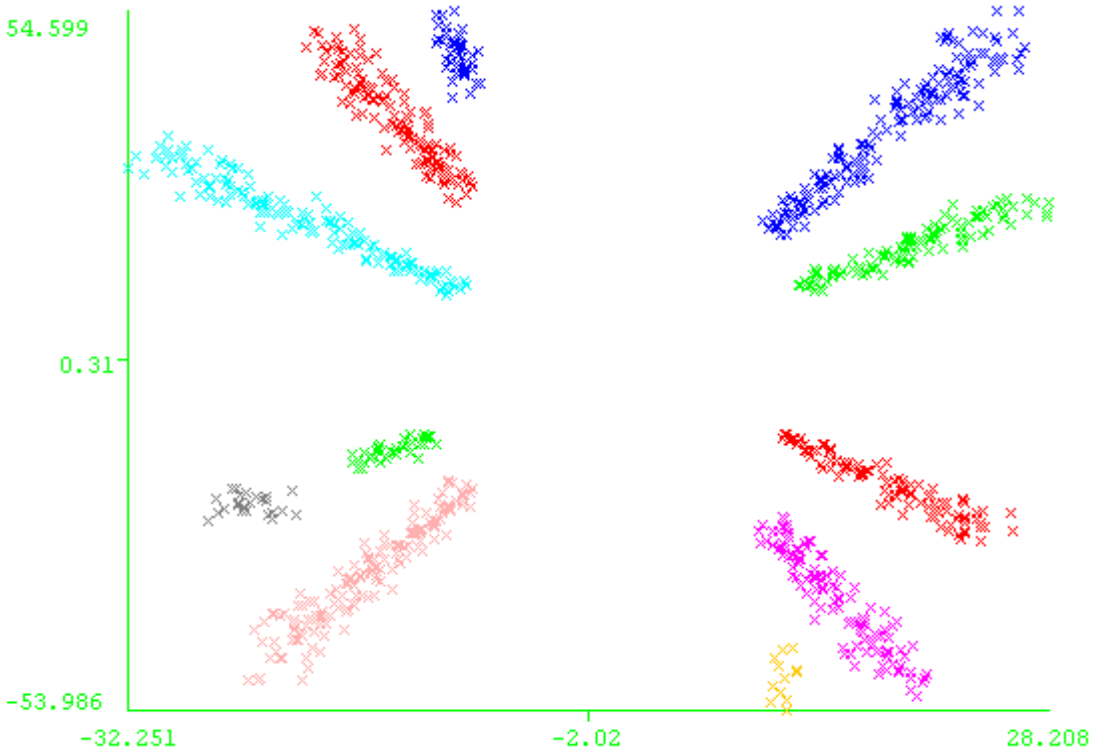
Come si nota per CobWeb e K-means infatti i quali non riescono a riconoscere forme complesse , l'algoritmo statistico riesce ad unire la semplicità del concetto spaziale dovuto alle regioni con gli studi l'efficenza di un algoritmo density Based quale DB scan , che comunque riporta prestazioni inferiori con tempi di calcolo superiori.

Test Galaxy

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 1068
- Numero di cluster → 11 senza rumore

Risultati :

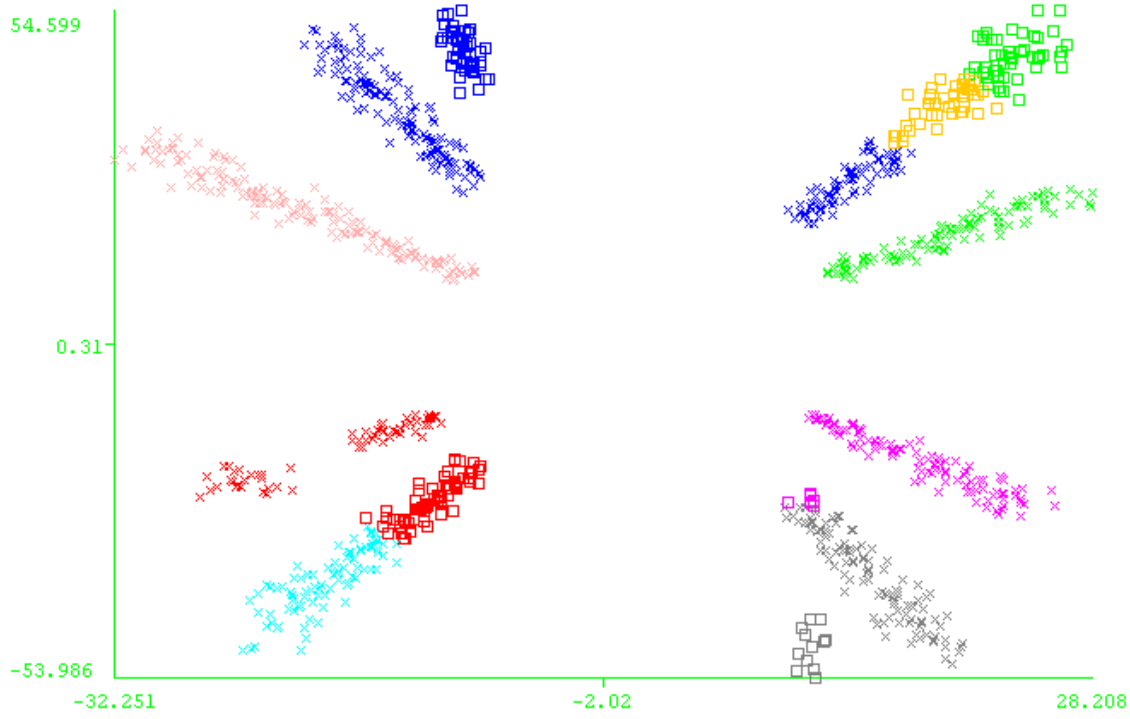


- Clustering statistico versione 2 → N=48 Confidenza =0.7
  - Numero di istanze errate → 70 ovvero il 6,55%
- Clustering statistico versione 3 → N=48 Confidenza =0.25
  - Numero di istanze errate → 7 ovvero il 0,70%
- DBSCAN E 0.035 MinPts 4
  - Numero di istanze errate → 1 ovvero il 0,10%
- CobWeb → A 6.22 C 0.0022
  - Numero di istanze errate → 175 ovvero il 14,67%
- Kmeans → Seed =44545 NumCluster = 11

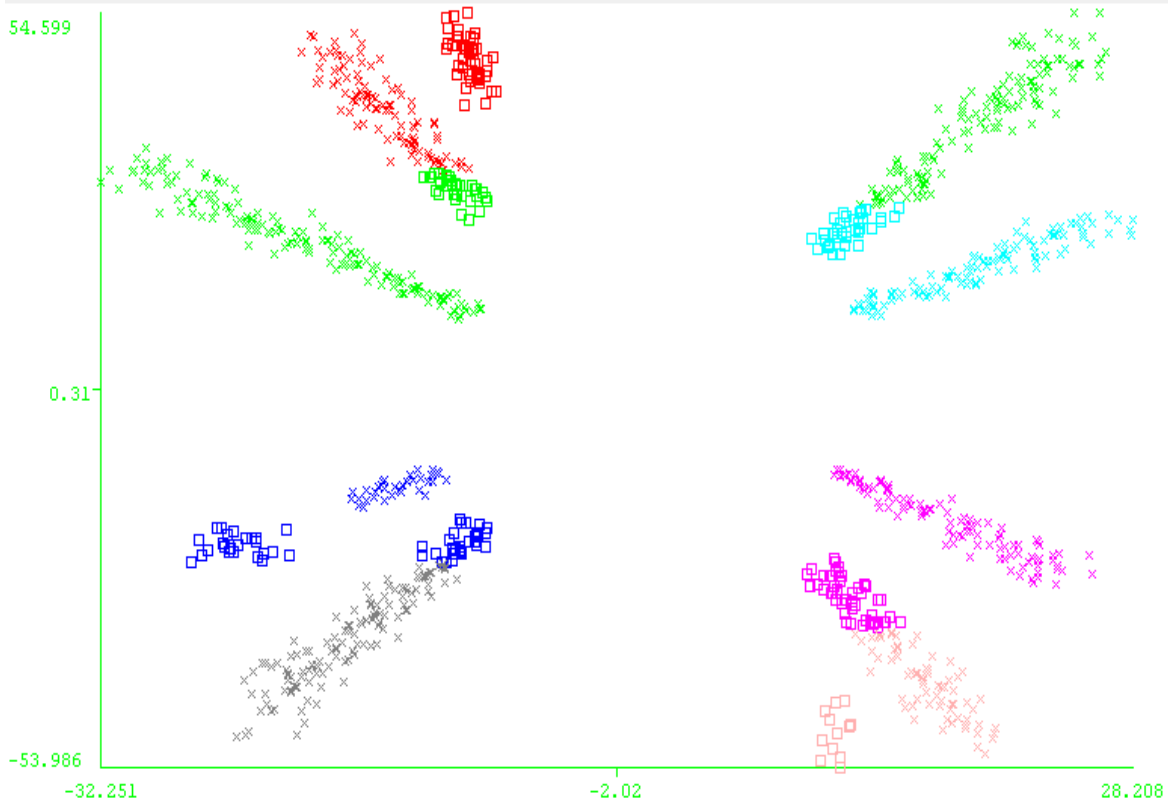
- Numero di istanze errate → 224 ovvero il 20,87%

Risultati Grafici Degli algoritmi

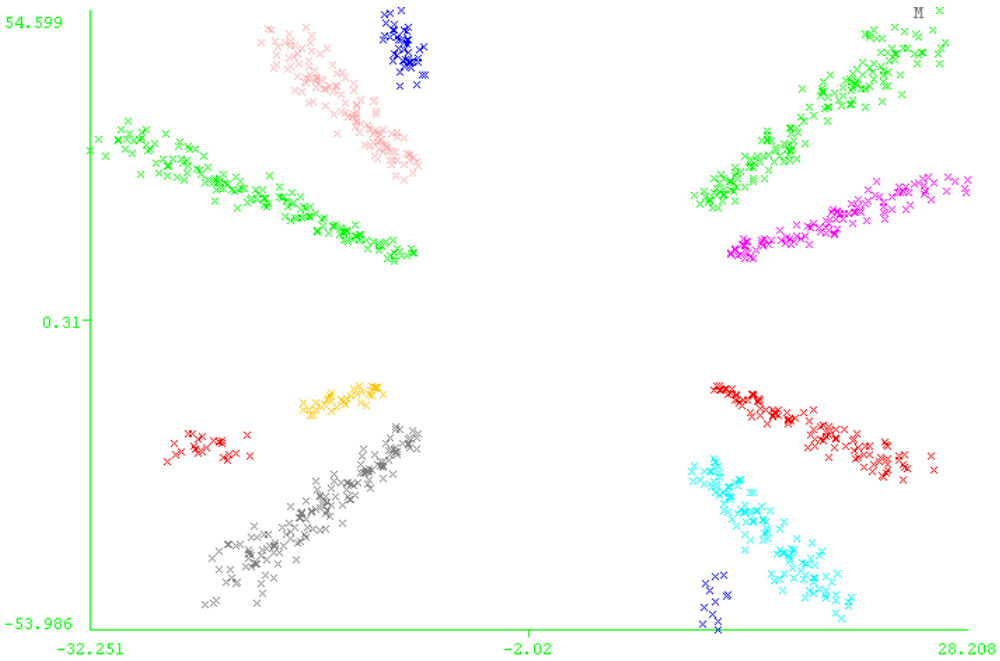
K-means



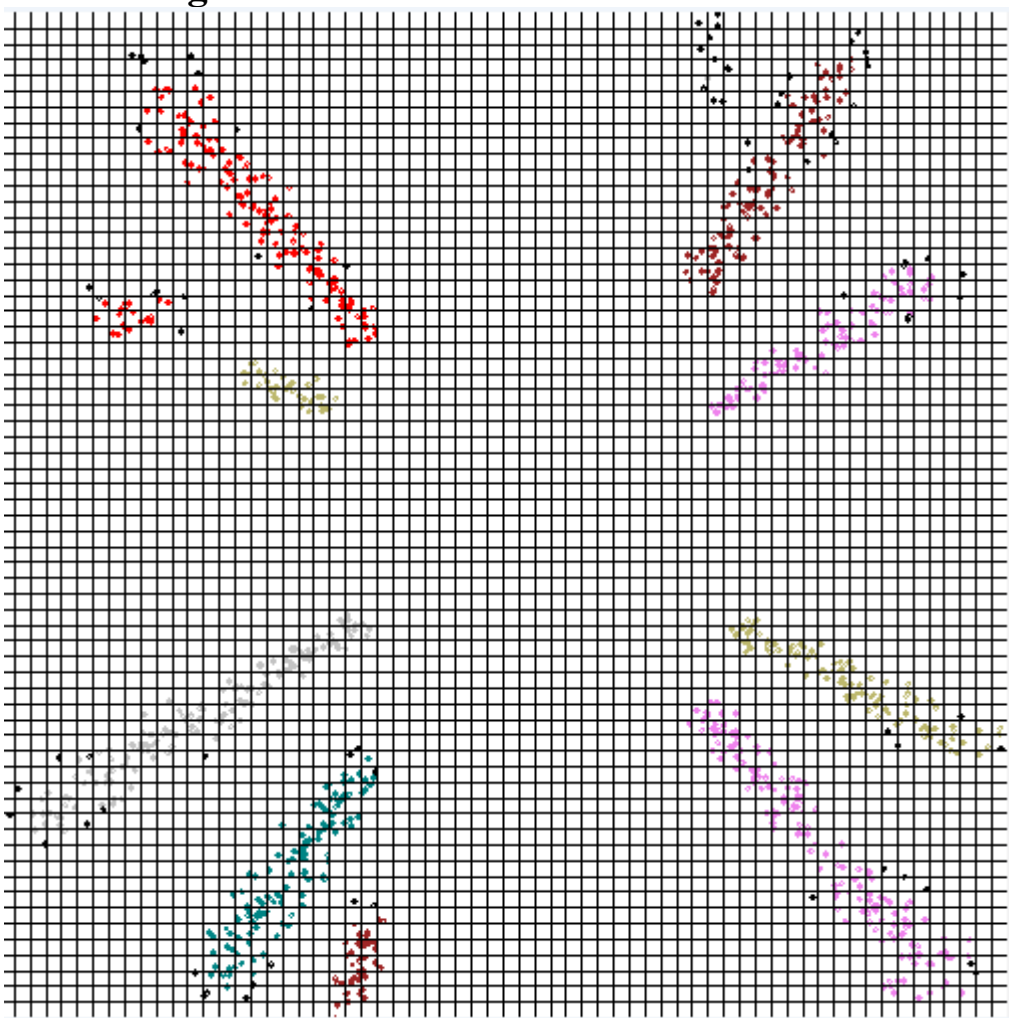
CobWeb



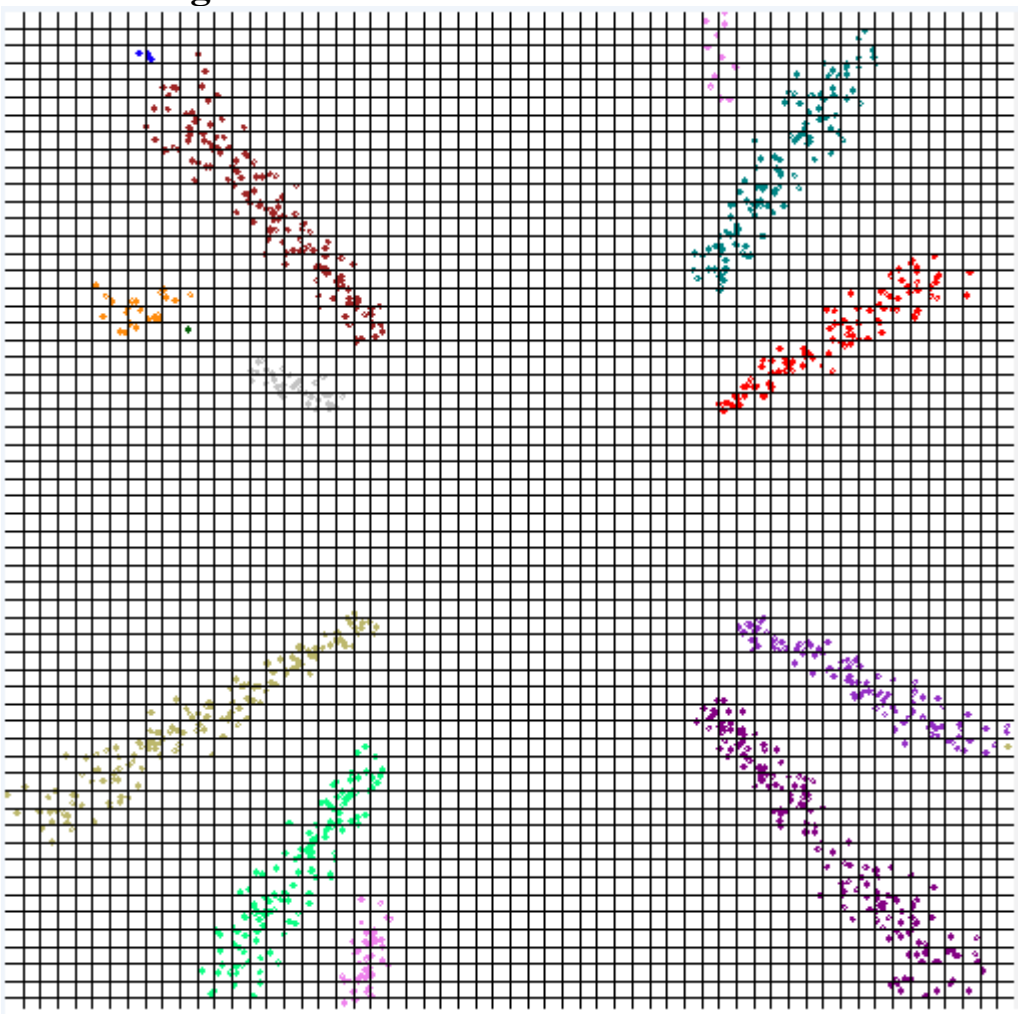
DB scan



ClusteringStatistico V 2.0



Clustering statistico 3.0



Considerazioni

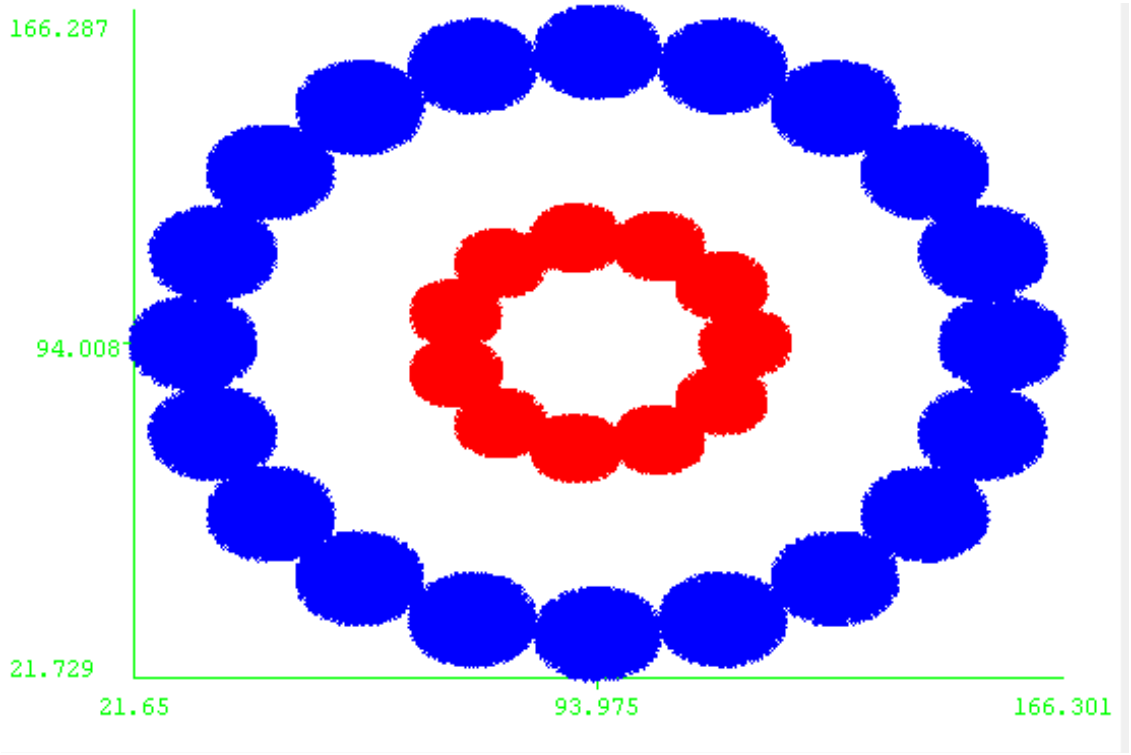
Dato il secondo dataset più semplice del precedente, ove non vi è affatto rumore ma i cluster sono posizionati in obliquo , si nota come entrambi gli algoritmi performino bene , leggermente meglio la versione 3 in quanto più “stringente” nell'identificare regioni casuali .

Test GirandBisBis

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 50.650
- Numero di cluster → 2 senza rumore

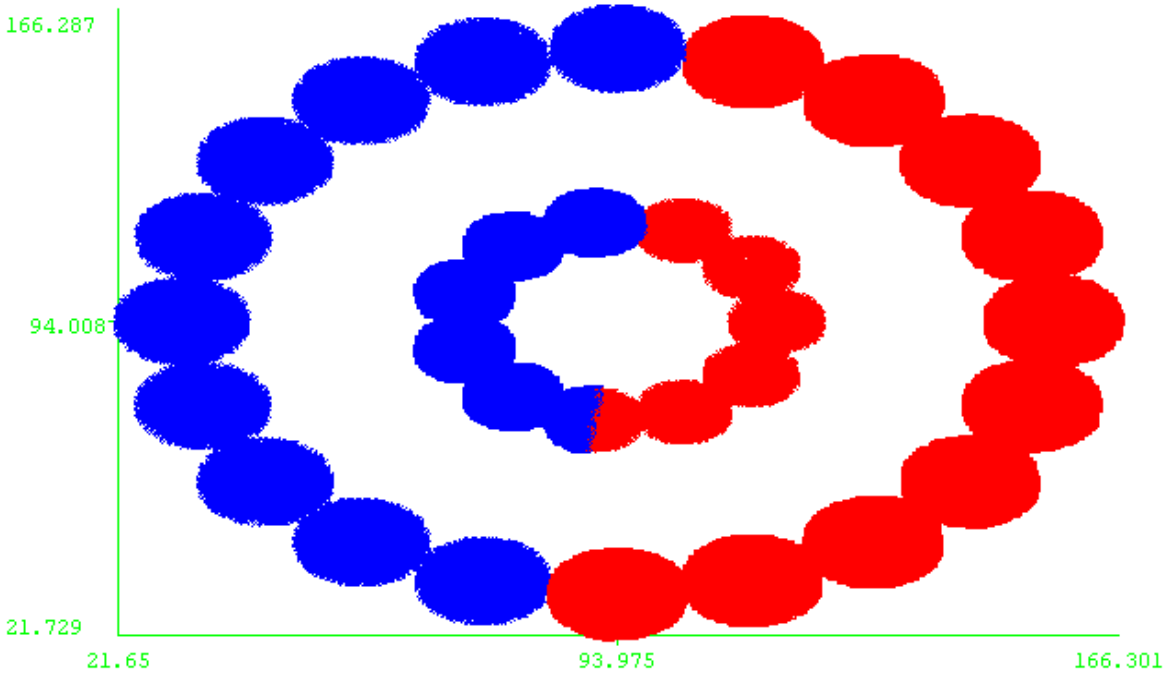
Risultati :



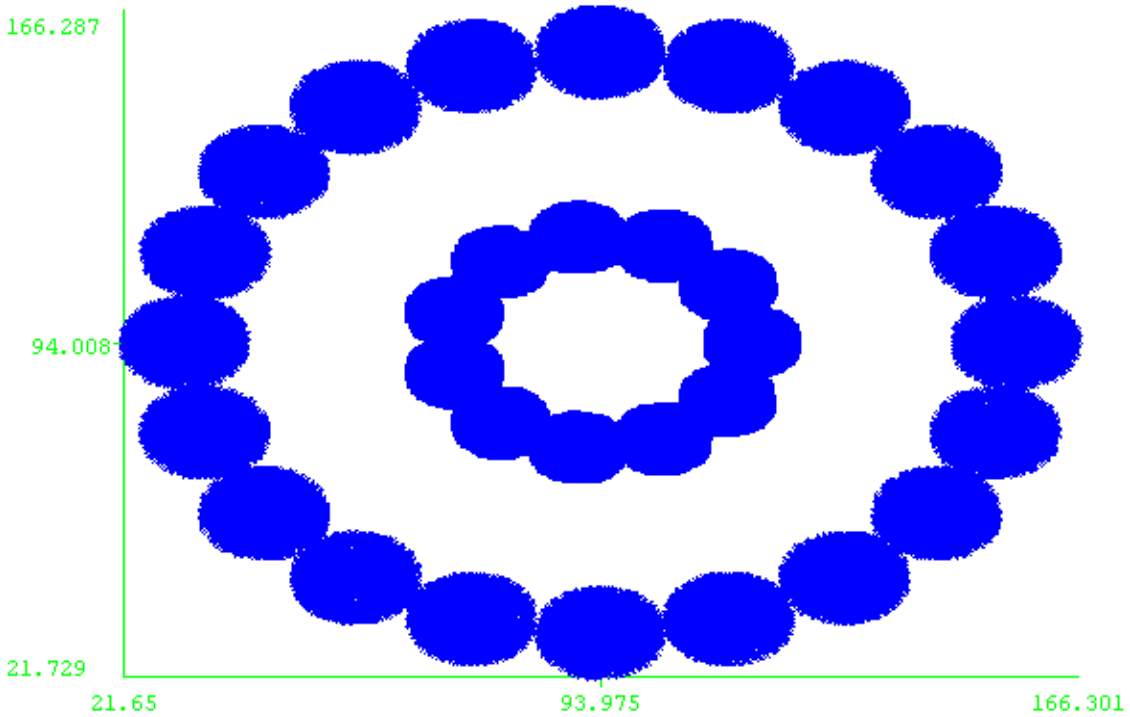
- Clustering statistico versione 2 → N=80 Confidenza =0.7
  - Numero di istanze errate → 888 ovvero il 1,38%
- Clustering statistico versione 3 → N=80 Confidenza =0.25
  - Numero di istanze errate → 888 ovvero il 1,38%
- DBSCAN E 0.02 MinPts 10
  - Numero di istanze errate → 0
- CobWeb → A 16.00 C 0.0037
  - Numero di istanze errate → 17230 ovvero il 26,84%
- Kmeans → Seed =445 NumCluster = 2
  - Numero di istanze errate → 32027 ovvero il 49,68%

Risultati Grafici Degli algoritmi

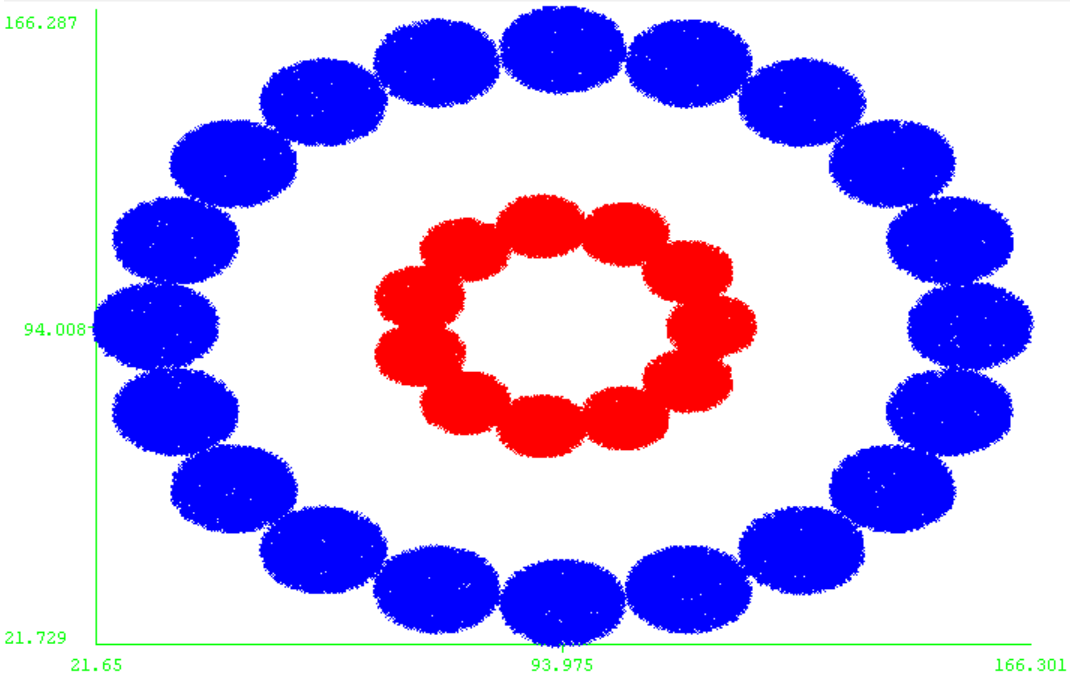
K-means



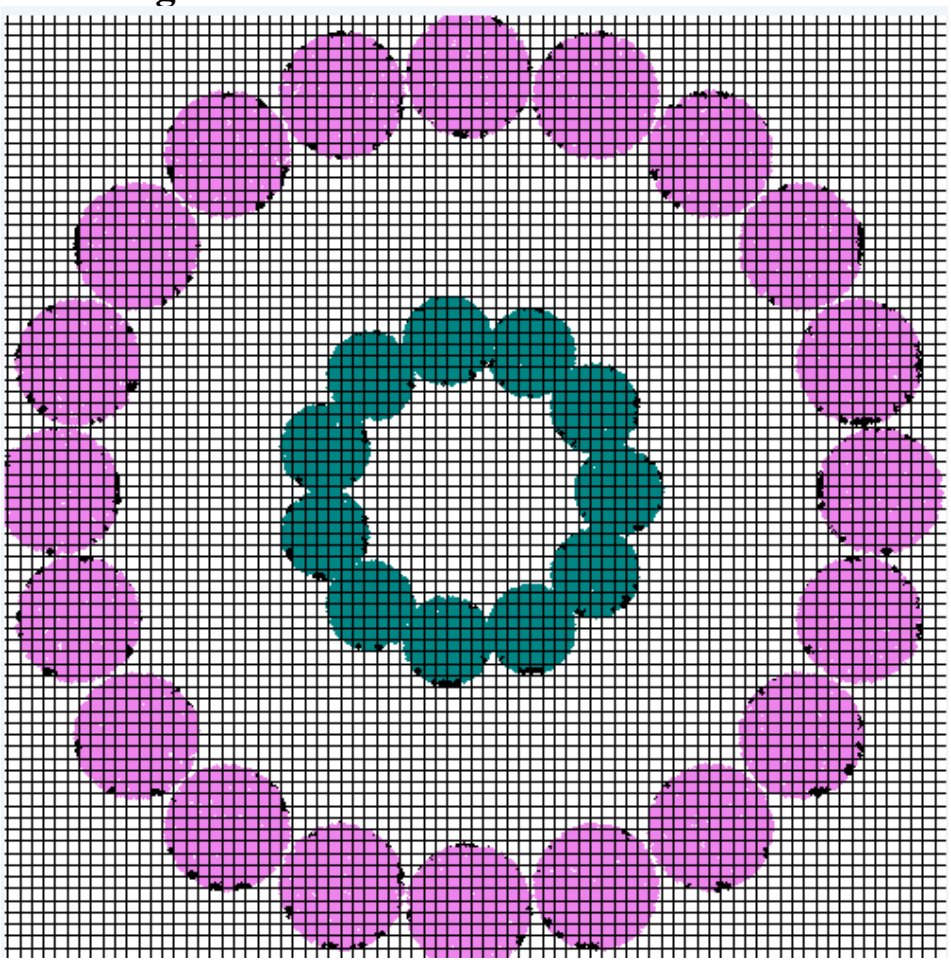
CobWeb



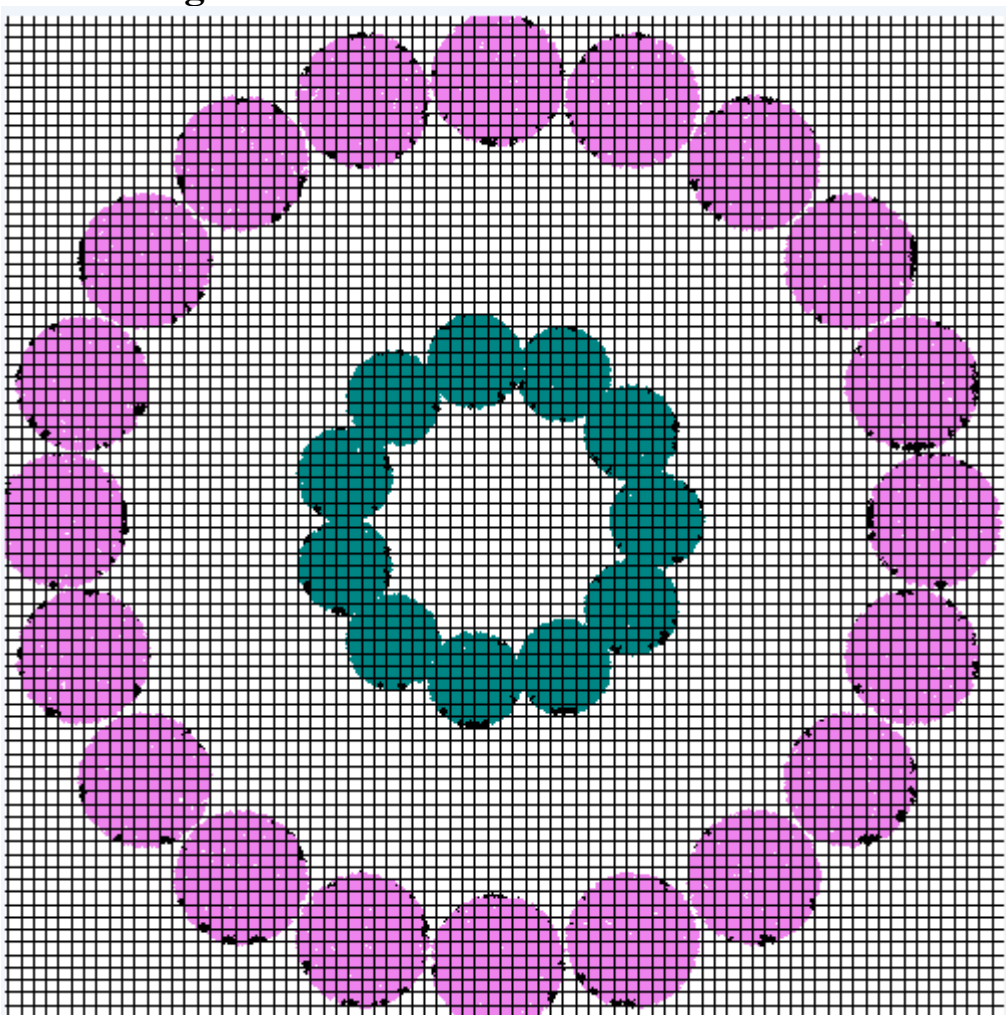
DB scan



ClusteringStatistico V 2.0



Clustering statistico 3.0



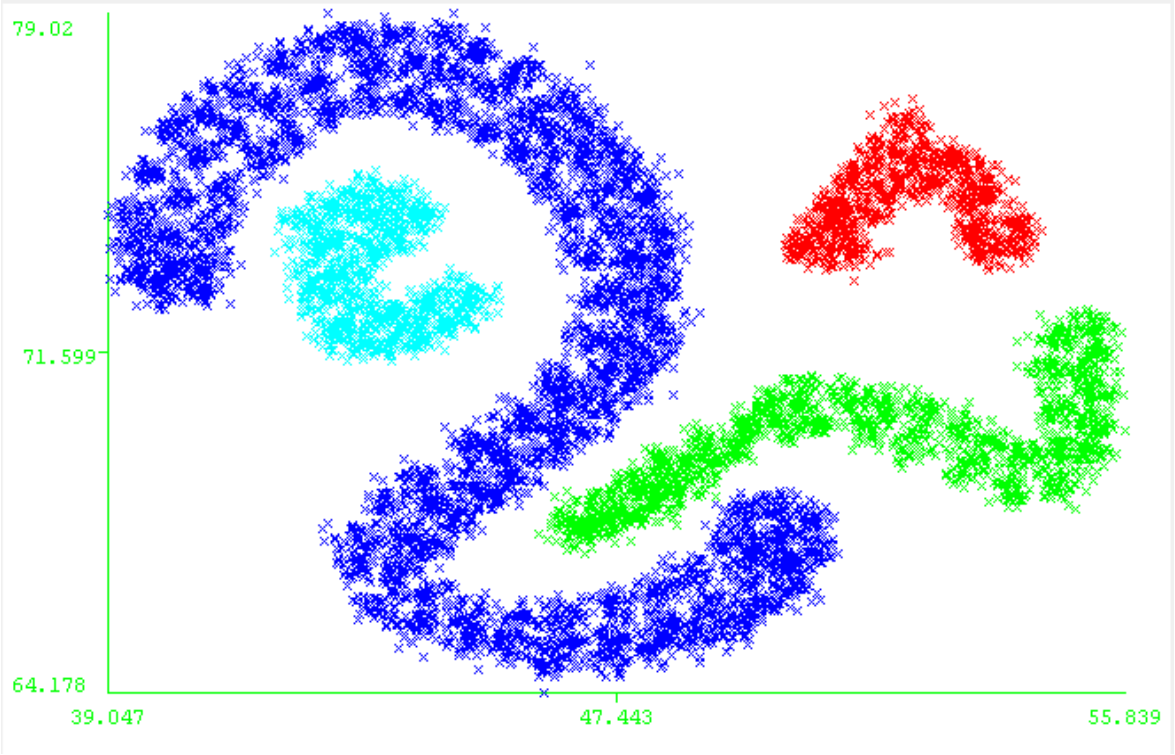
Considerazioni

Il dataset presenta una elevata numerosità , ma data la suddivisione a griglia anche con N elevato (80 corrisponde a 6400 regioni) vi è sempre una relazione 1 a 10 tra regioni e punti .
Questo permette all'algoritmo di restituire un ottimo output in tempi brevi (qualche secondo) rispetto ad algoritmi orientati ai punti quali DBSCAN che impiegano sui 30 minuti.
In questo caso tutti gli algoritmi a meno di CobWeb e K-means hanno portato ad eccellenti risultati, un appunto su K-means è però doveroso ; difatti impostando il numero esatto di cluster che ci si aspetta si agevola molto il compito dell'algoritmo stesso ; quindi risulterebbe meno efficace in un caso di utilizzo reale.

Test ex2m50

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 10.000
- Numero di cluster → 4 senza rumore

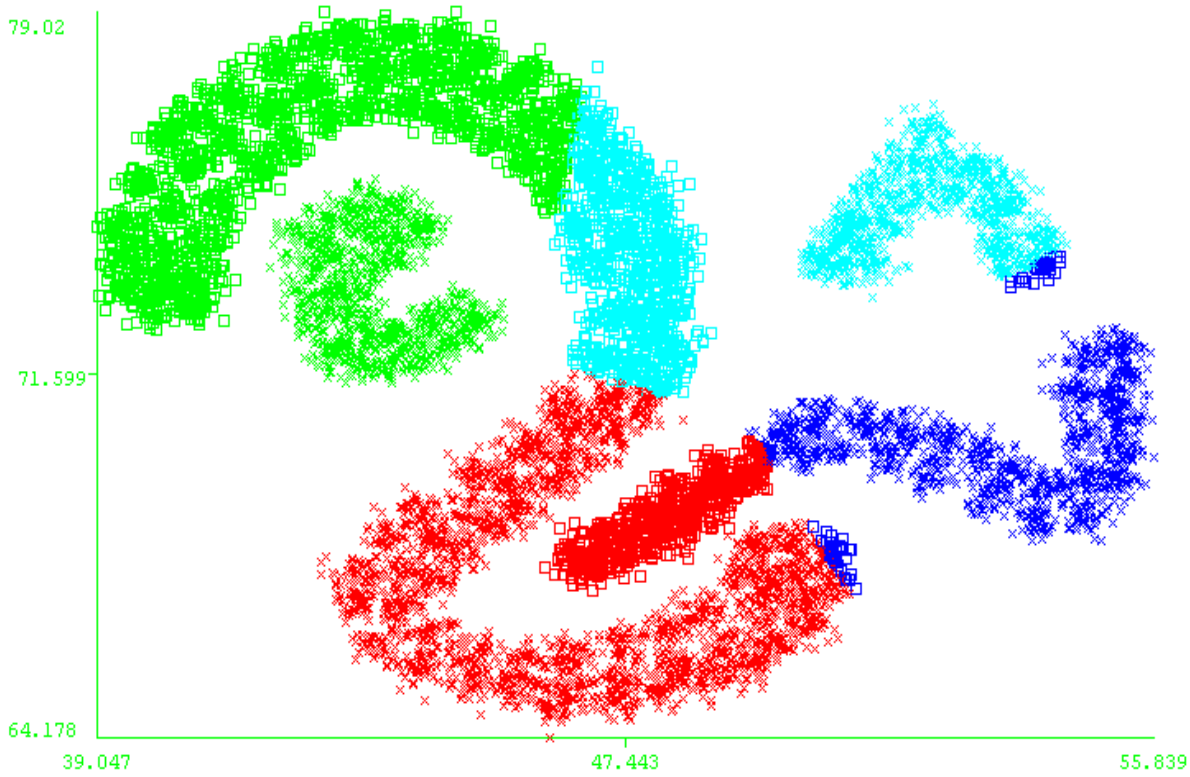


Risultati :

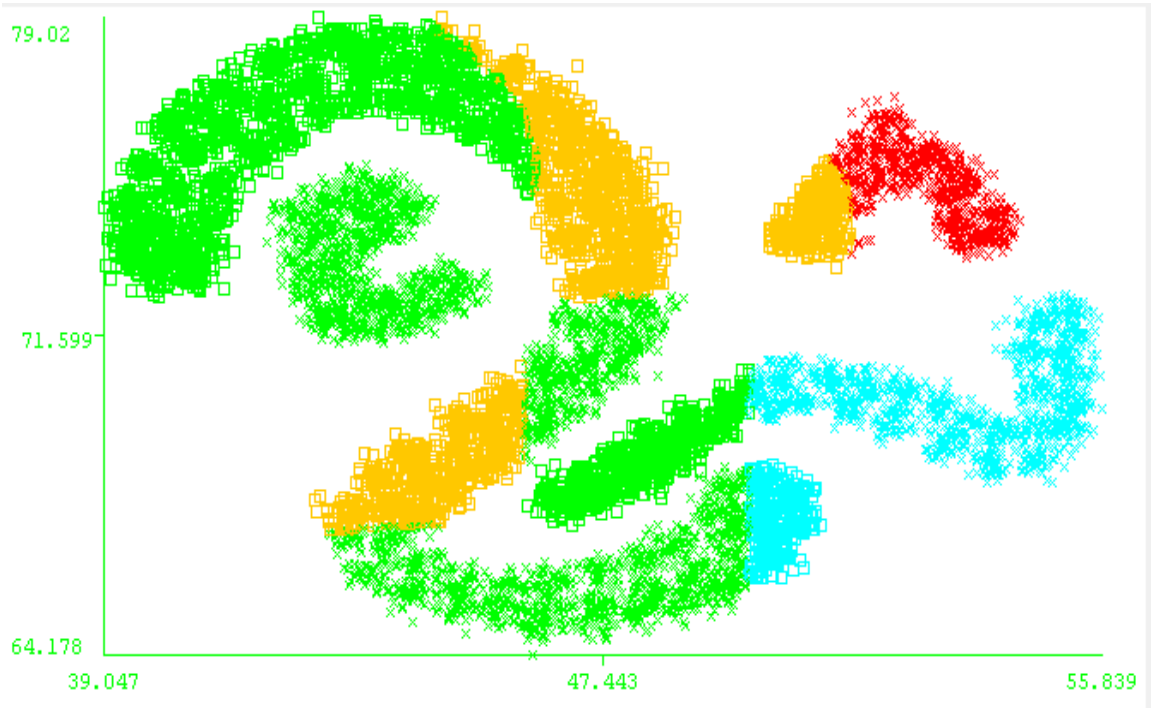
- Clustering statistico versione 2 → N=46 Confidenza =0.7
  - Numero di istanze errate → 309 ovvero il 3,09%
- Clustering statistico versione 3 → N=46 Confidenza =0.25
  - Numero di istanze errate → 309 ovvero il 3,09%
- DBSCAN E 0.015 MinPts 6
  - Numero di istanze errate → 42 ovvero il 0,50%
- CobWeb → A 2.0 C 0.0021
  - Numero di istanze errate → 4799 ovvero il 48,49%
- Kmeans → Seed =446 NumCluster = 4
  - Numero di istanze errate → 3819 ovvero il 38,19%

Risultati Grafici Degli algoritmi

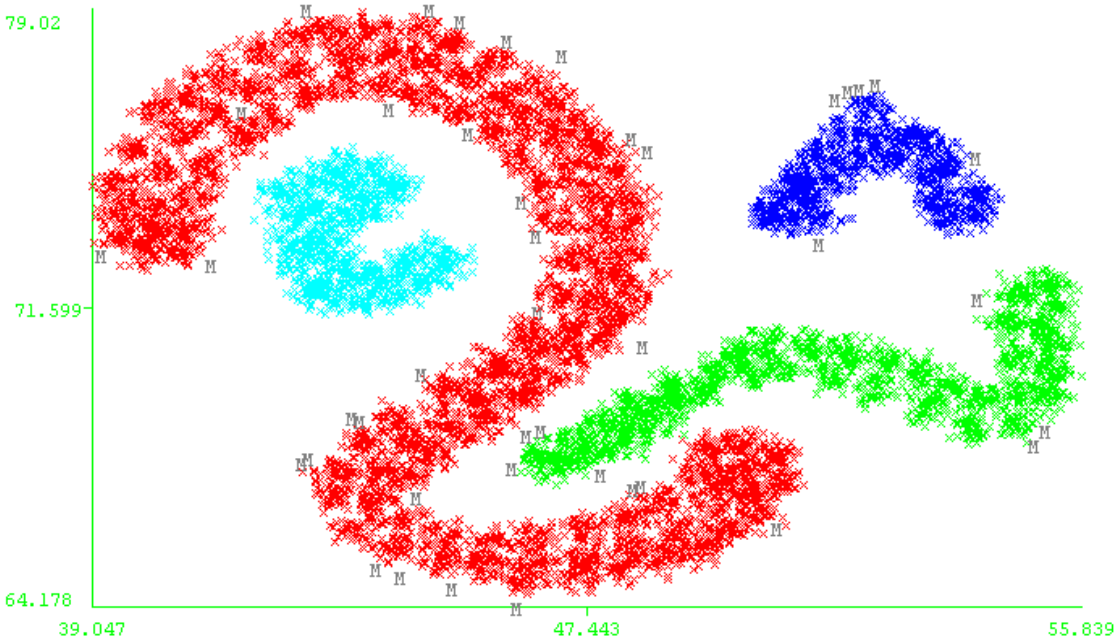
K-means



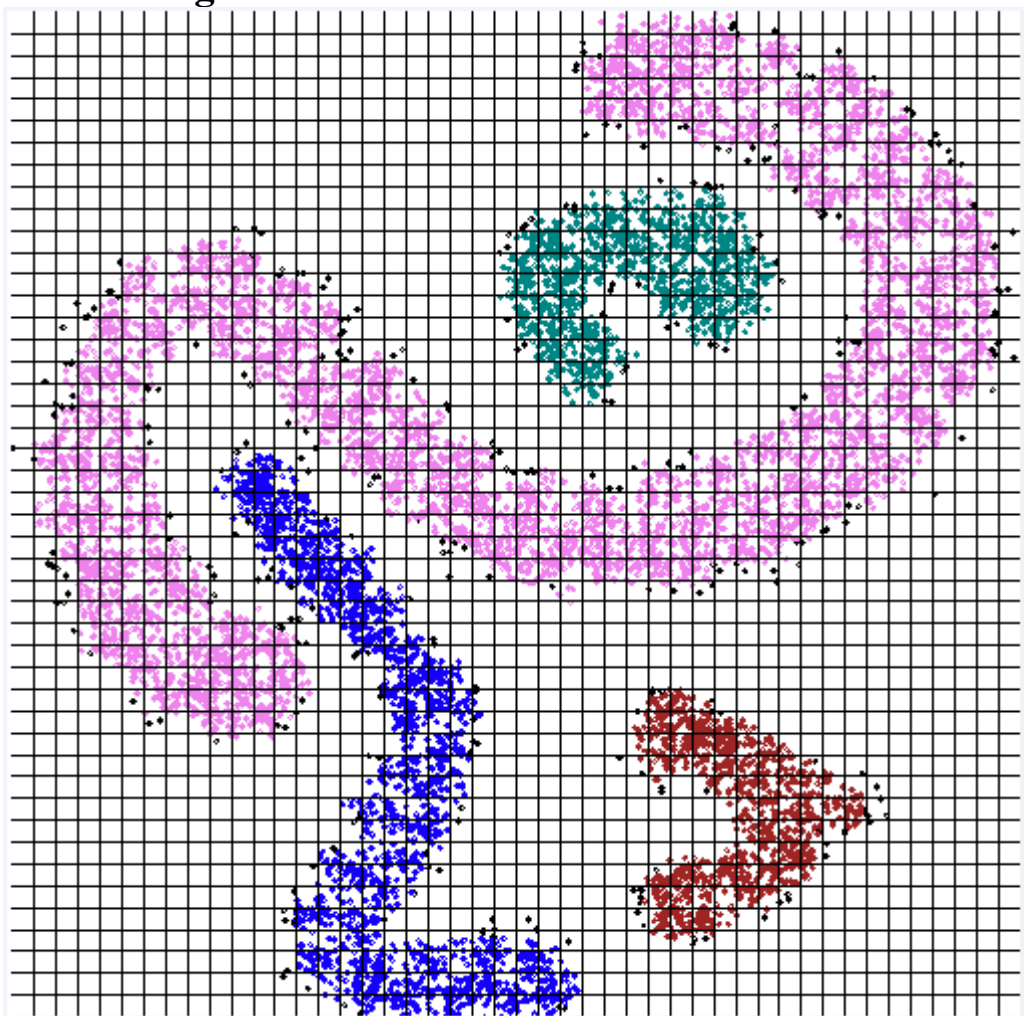
CobWeb



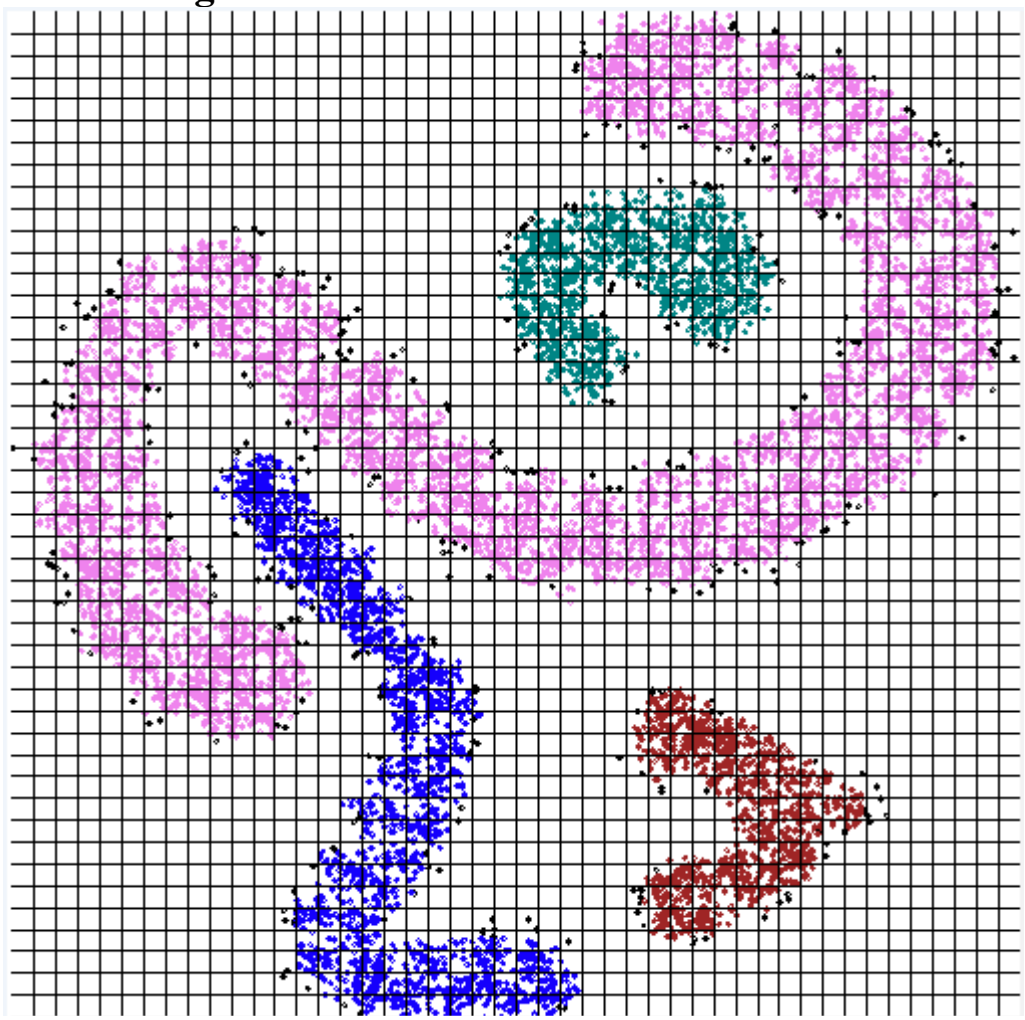
- **DB scan**



ClusteringStatistico V 2.0



Clustering statistico 3.0



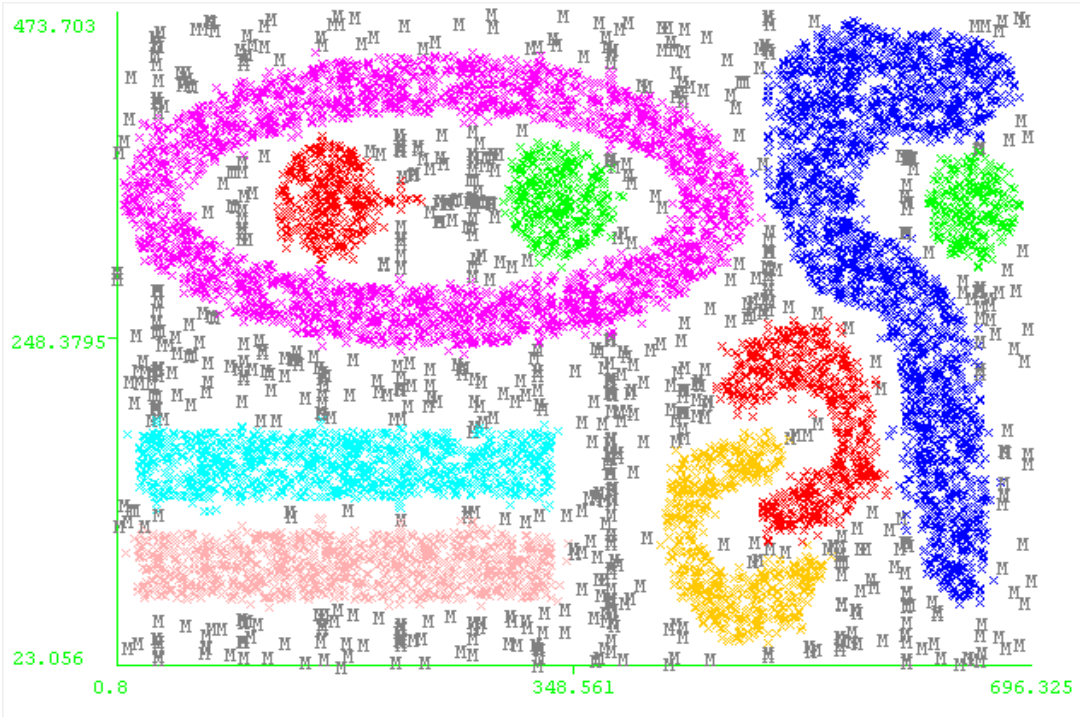
Considerazioni

Le forme dei cluster sono non lineari e questo mette in difficoltà algoritmi quali Kmeans e CobWeb .
Come per il primo test anche in questo caso entra in gioco la corretta suddivisione delle celle .
Non vi è alcun problema nella divisione tra regioni Dense + / - e regioni casuali, mentre l'algoritmo di clustering nella sua semplicità lascia indietro i punti “margine” del cluster in quanto appartenenti a regioni a bassa numerosità.

Test T7.10K

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 10.000
- Numero di cluster → 9 con rumore

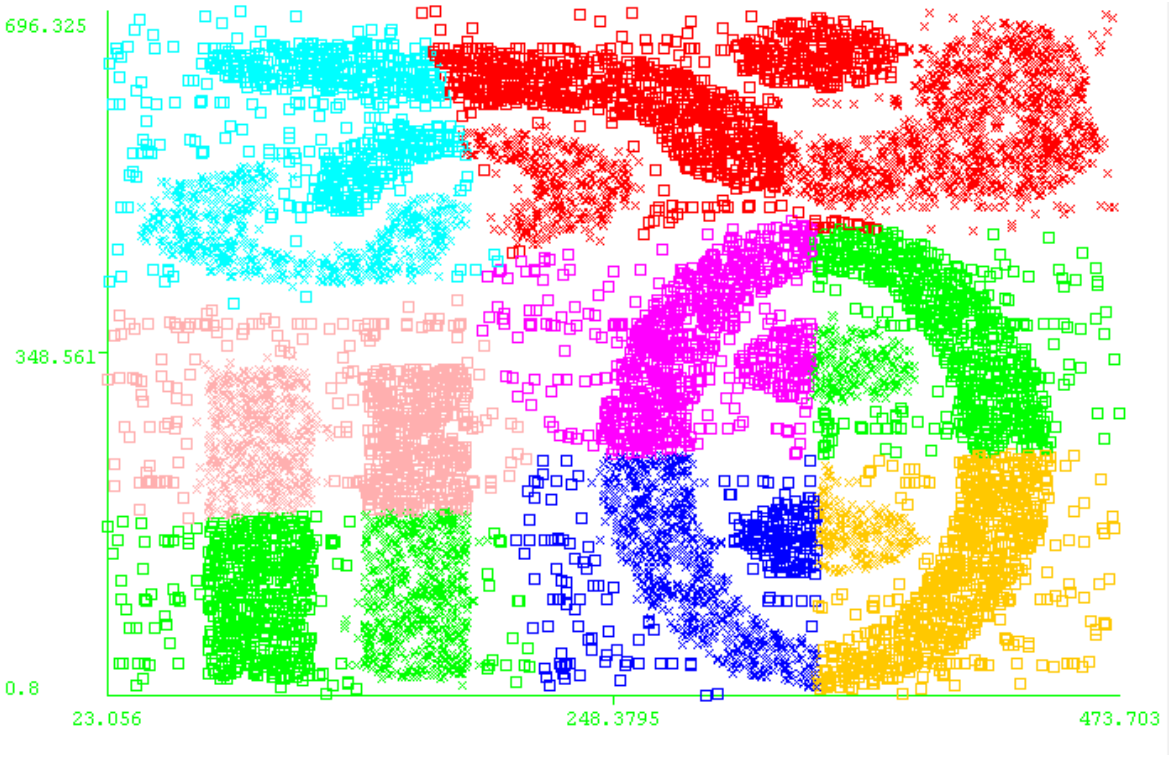


Risultati :

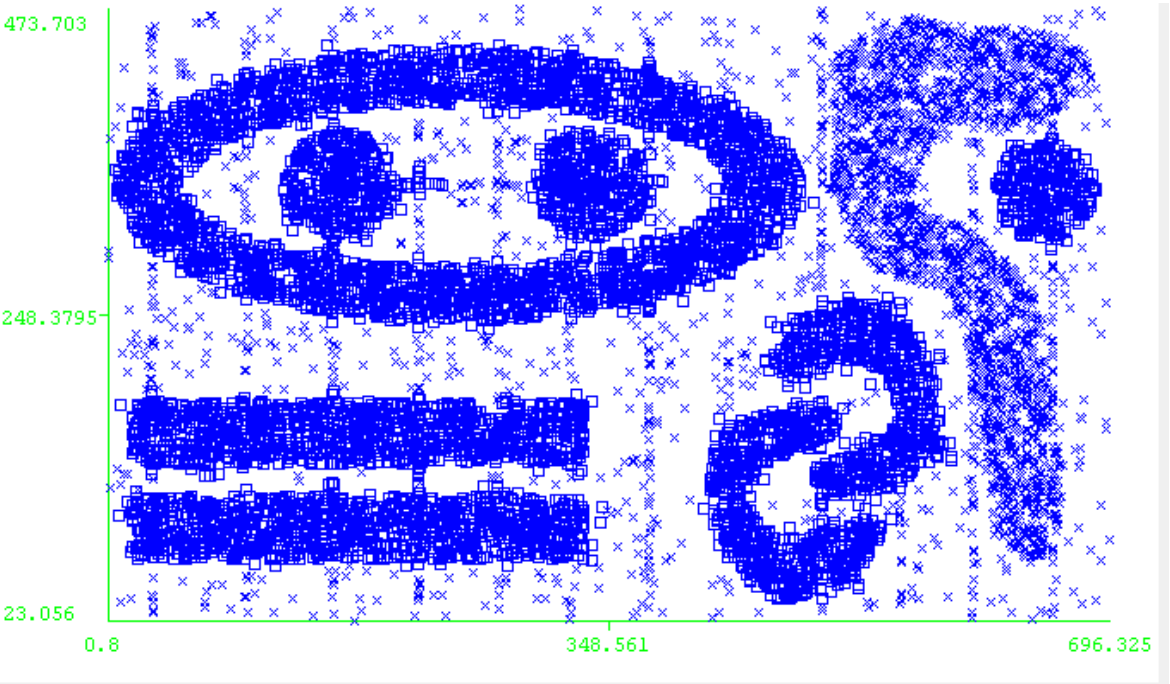
- Clustering statistico versione 2 → N=49 Confidenza =0.35
  - Numero di istanze errate → 909 ovvero il 9,09%
- Clustering statistico versione 3 → N=49 Confidenza =0.65
  - Numero di istanze errate → 890 ovvero il 8,9%
- DBSCAN E 0.015 MinPts 6
  - Numero di istanze errate → 488 ovvero il 4,88%
- CobWeb → A 2.0 C 0.0021
  - Numero di istanze errate → 7016 ovvero il 70,16%
- Kmeans → Seed =446 NumCluster = 4
  - Numero di istanze errate → 5630 ovvero il 56,3%

Risultati Grafici Degli algoritmi

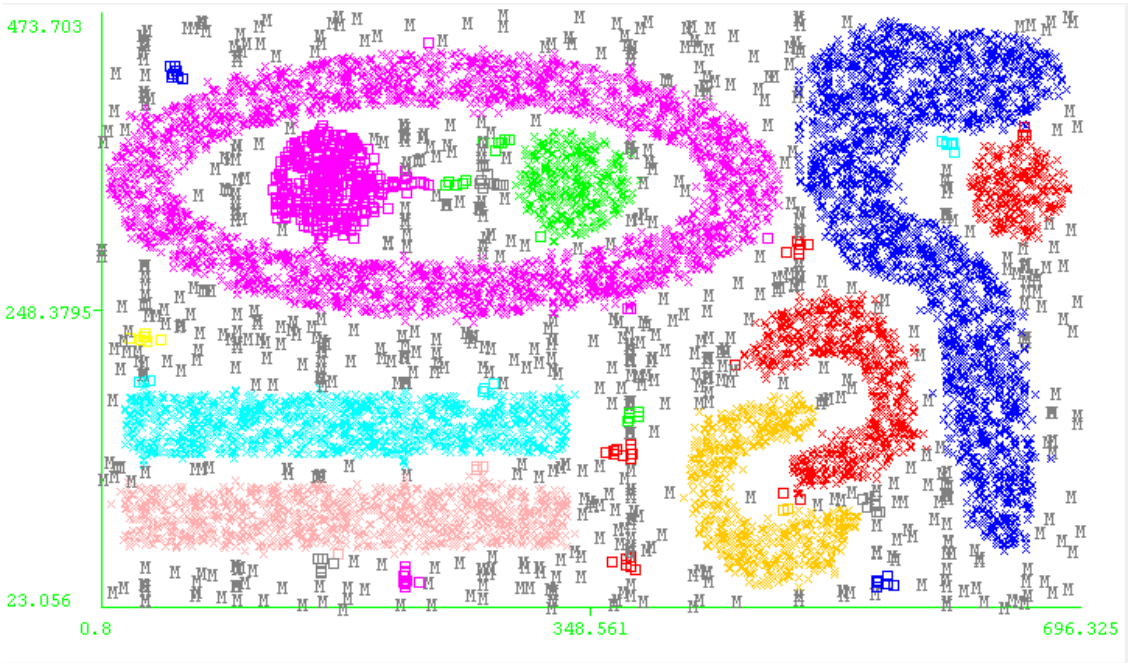
K-means



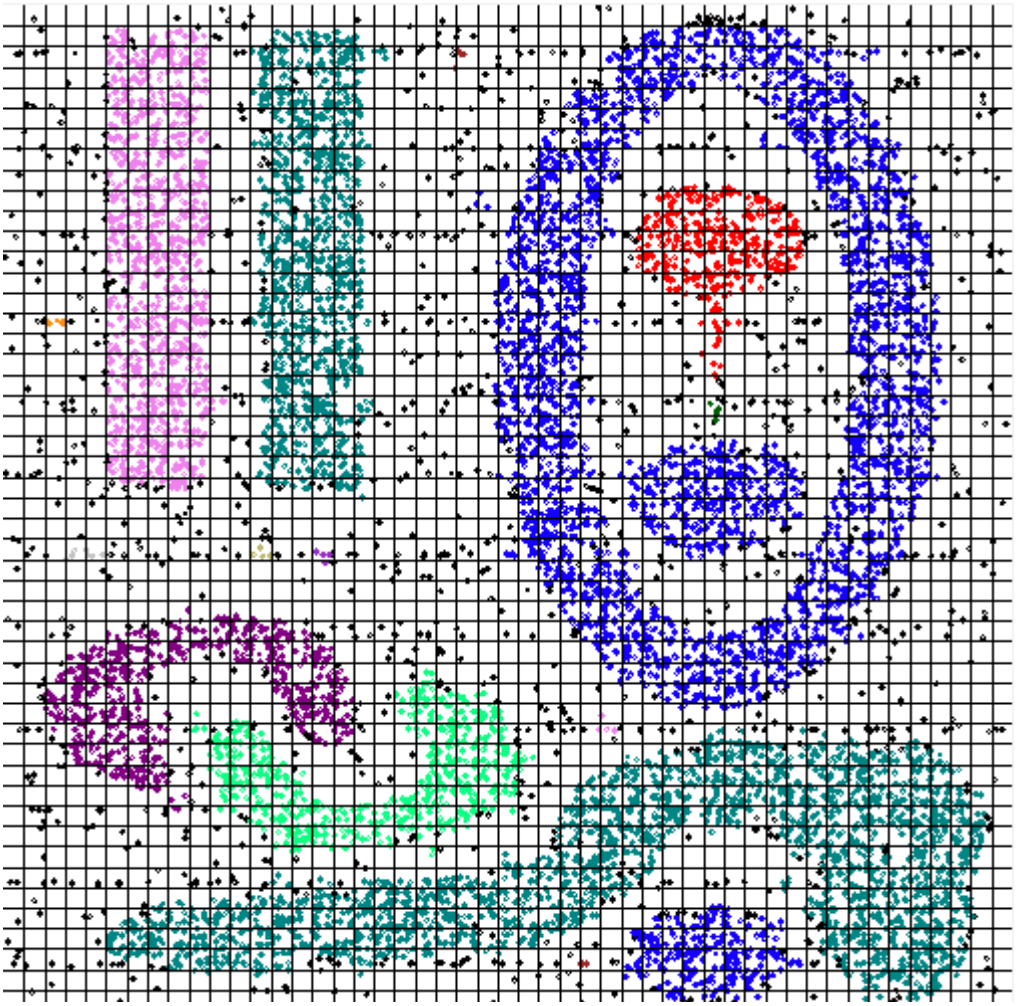
CobWeb



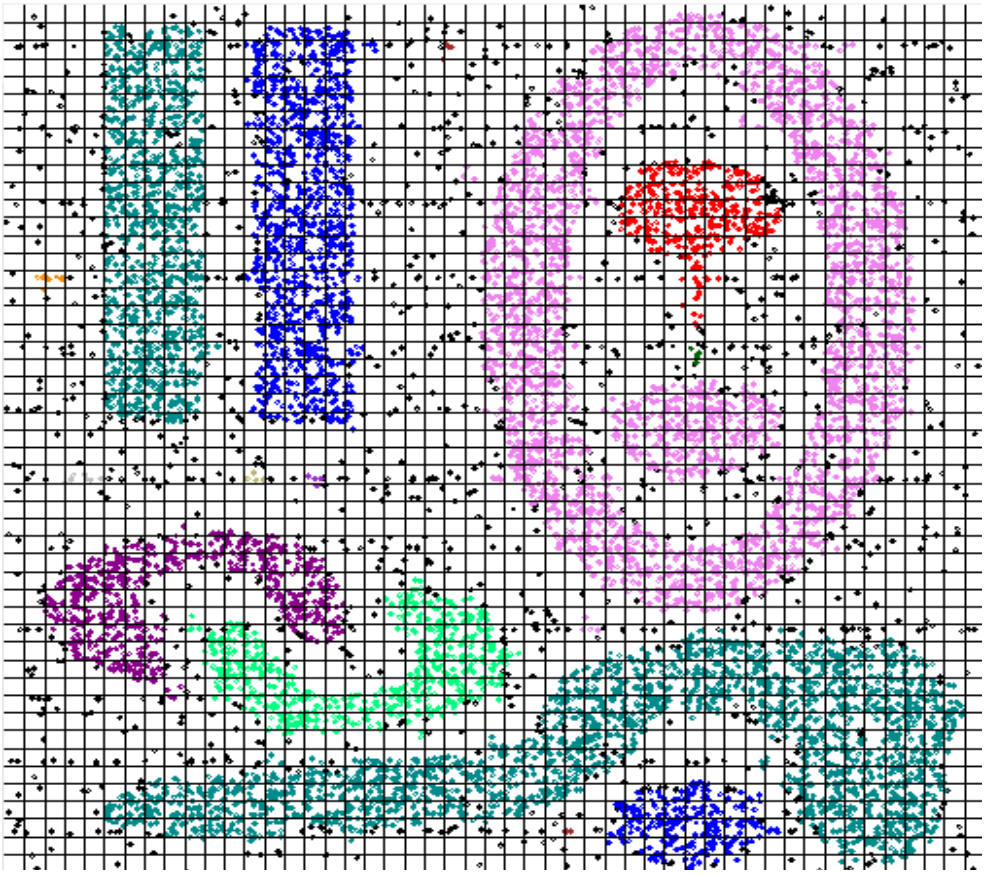
DB scan



ClusteringStatistico V 2.0



Clustering statistico 3.0



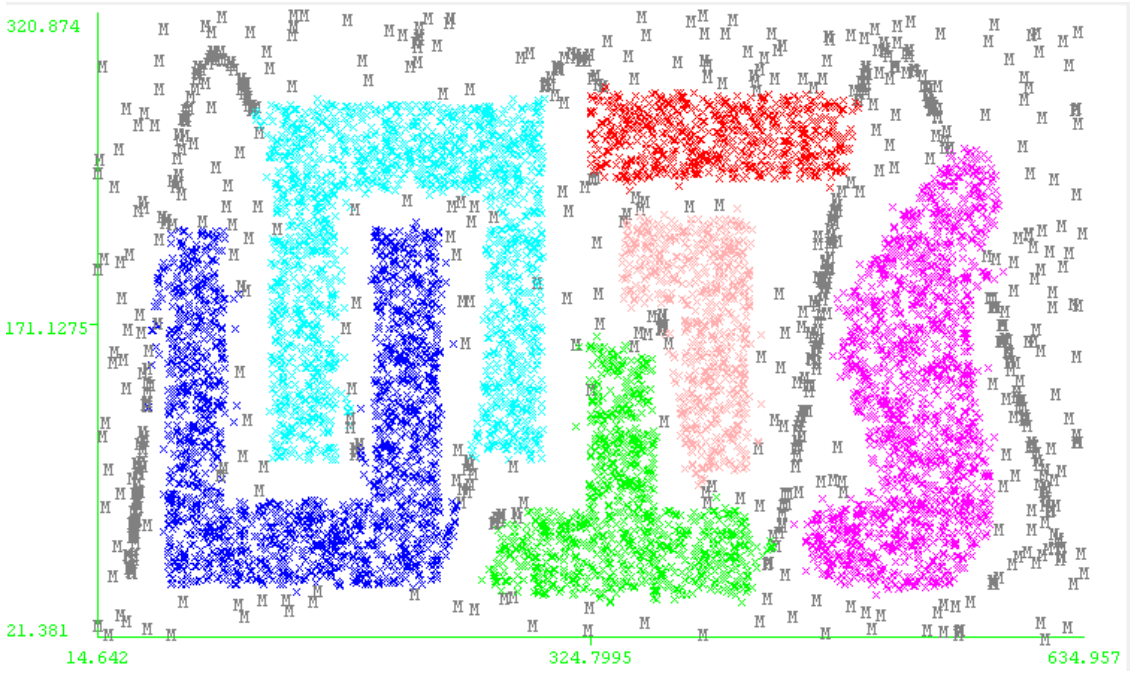
Considerazioni

Le forme dei cluster sono non lineari e questo mette in difficoltà algoritmi quali Kmeans e CobWeb .
Come per il primo test anche in questo caso entra in gioco la corretta suddivisione delle celle .
Dalle prove effettuate l'algoritmo 3.0 identifica chiaramente le zone casuali (i punti centrali di rumore) mentre il 2.0 le identifica come DENSE - .
La problematica invece del clustering ha fatto si che uno dei due cluster all'interno dell'ovale sia stato considerato come un solo cluster.

Test T4.8K

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 8.000
- Numero di cluster → 6 con rumore

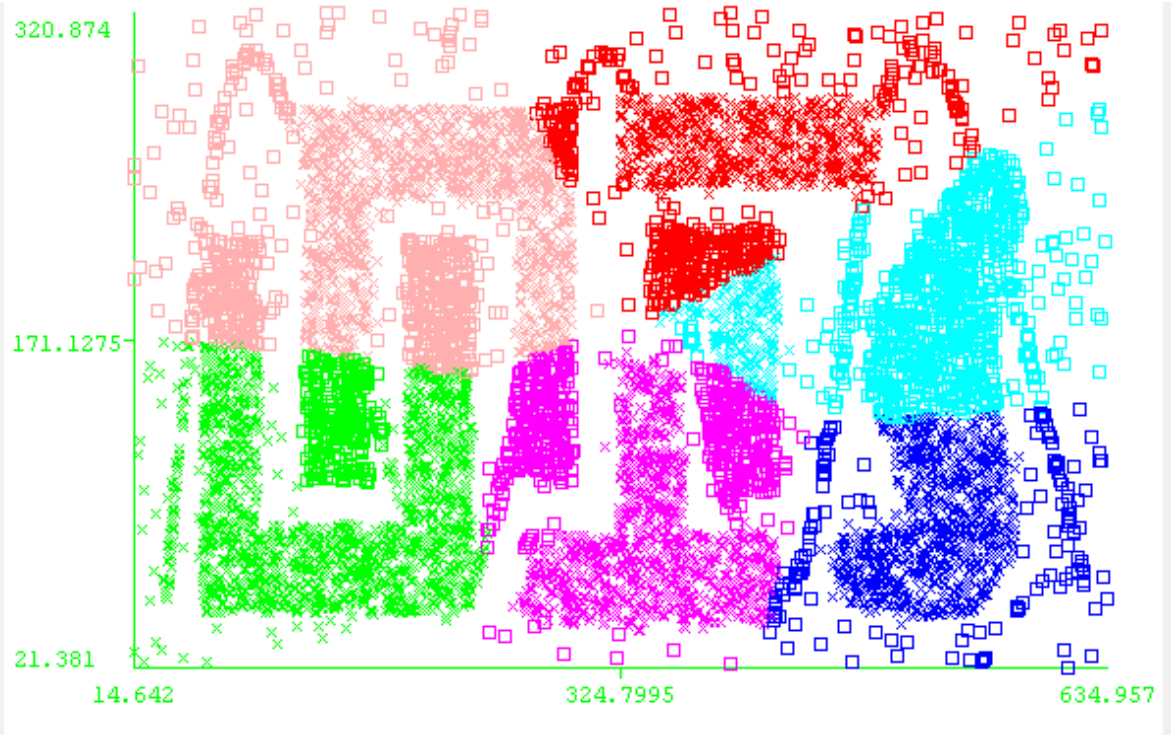


Risultati :

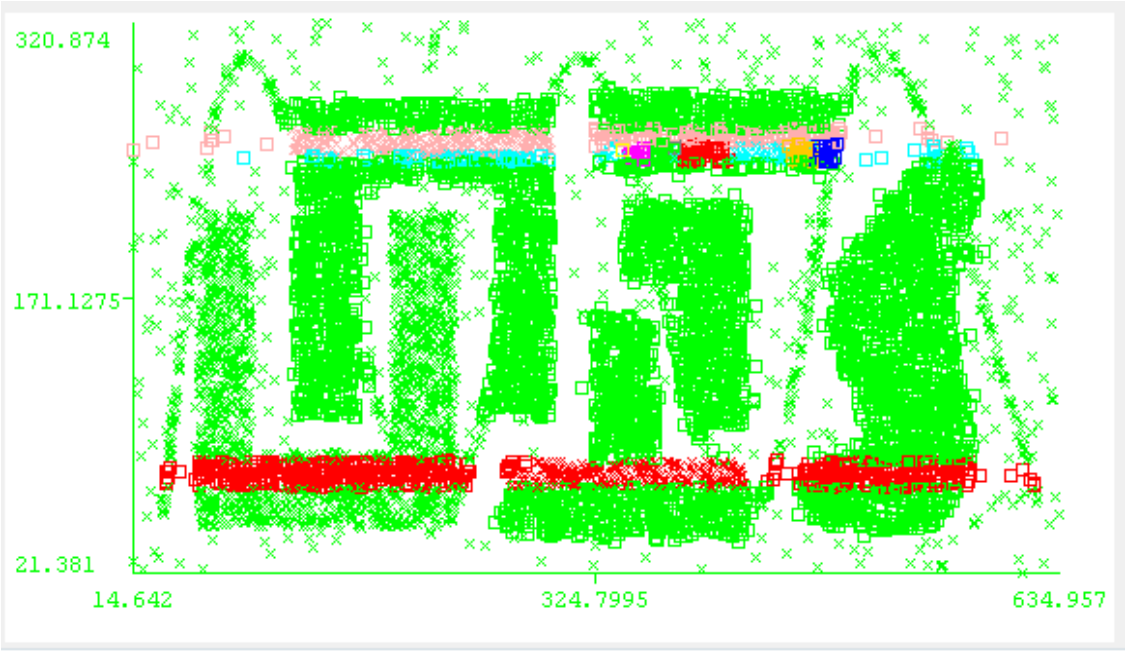
- Clustering statistico versione 2 → N=49 Confidenza =0.36
  - Numero di istanze errate → 1212 ovvero il 15,15%
- Clustering statistico versione 3 → N=49 Confidenza =0.65
  - Numero di istanze errate → 1168 ovvero il 14,6%
- DBSCAN E 0.015 MinPts 6
  - Numero di istanze errate → 252 ovvero il 3,15%
- CobWeb → A 2.0 C 0.0021
  - Numero di istanze errate → 5252 ovvero il 68,49%
- Kmeans → Seed =446 NumCluster = 4
  - Numero di istanze errate → 2737 ovvero il 34,21%

- **Risultati Grafici Degli algoritmi**

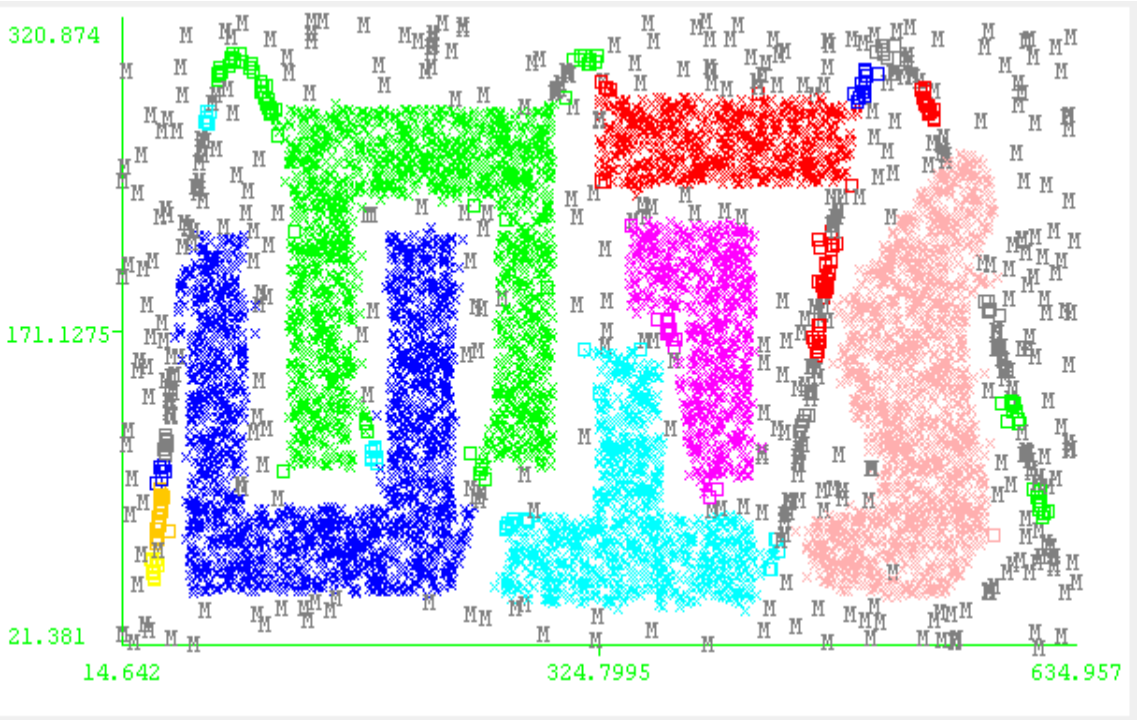
K-means



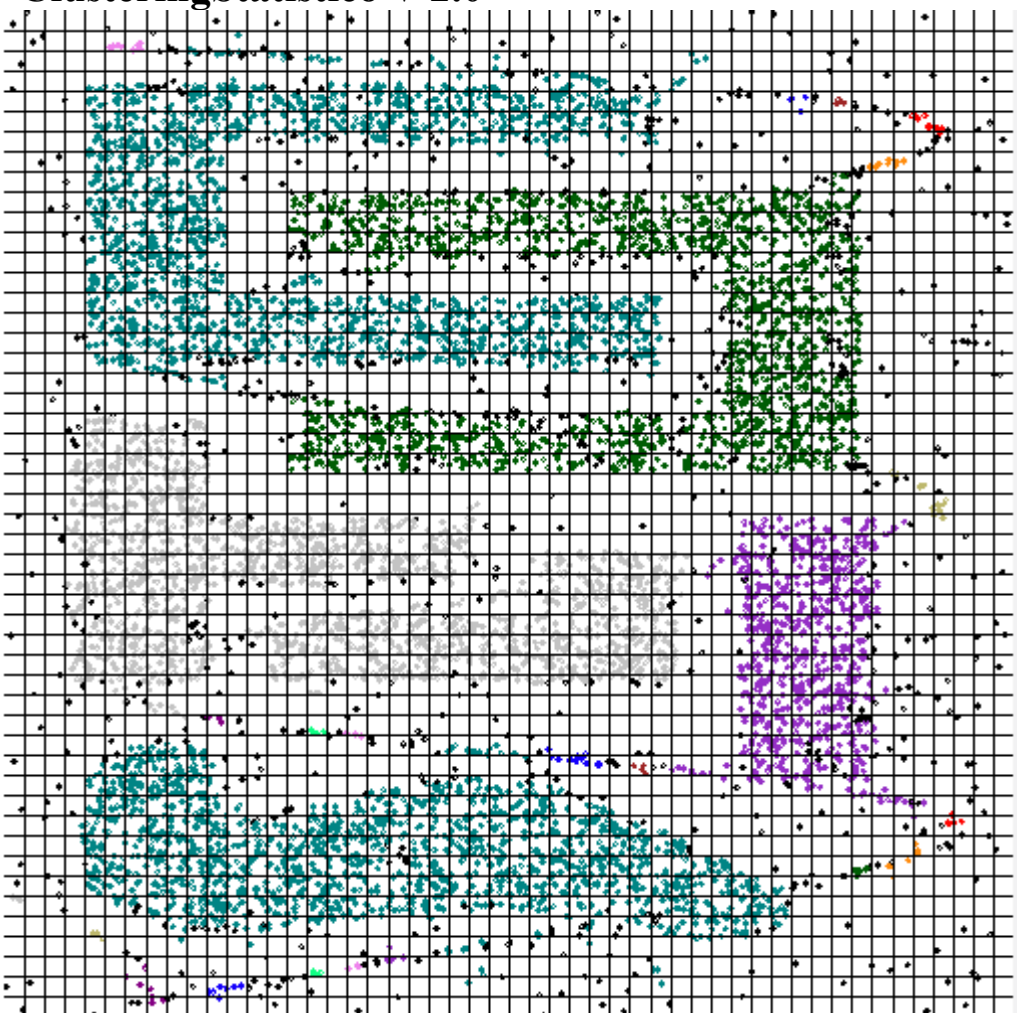
CobWeb



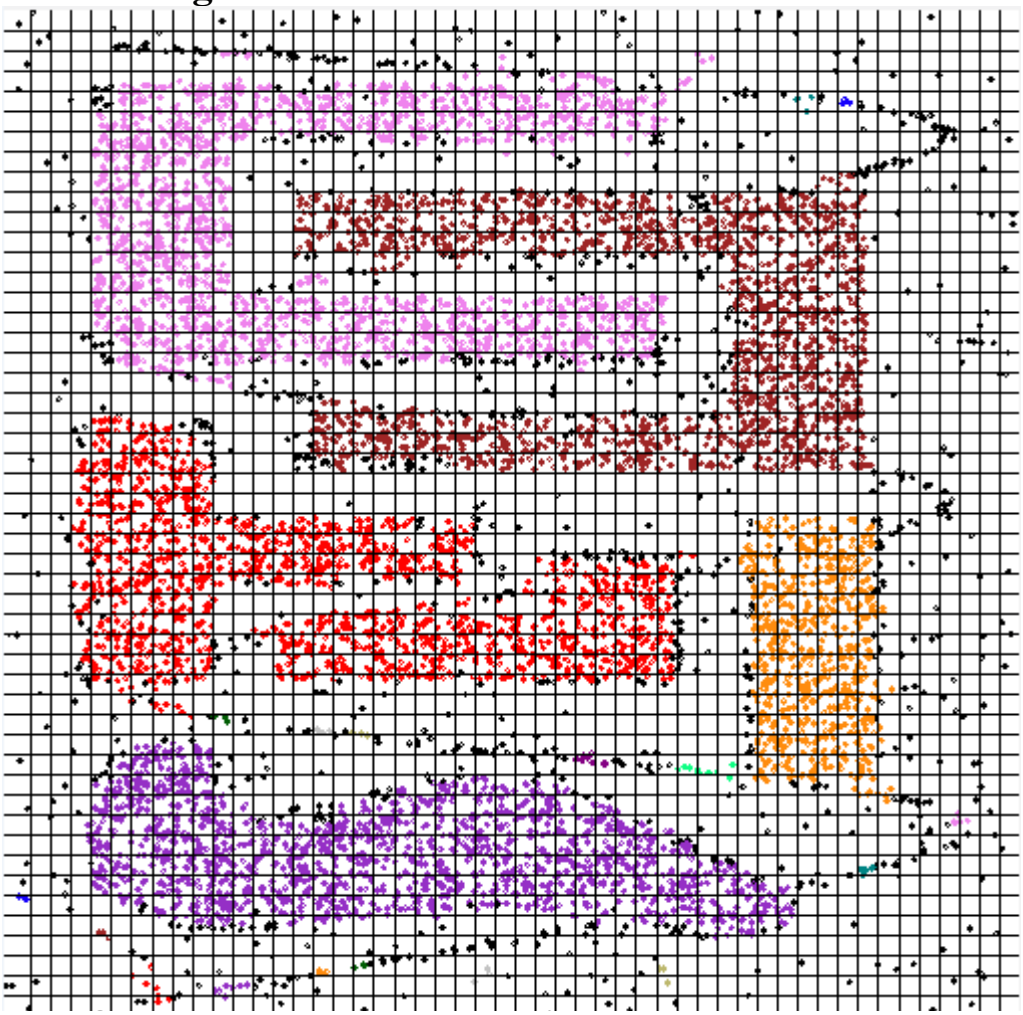
DB scan



ClusteringStatistico V 2.0



Clustering statistico 3.0



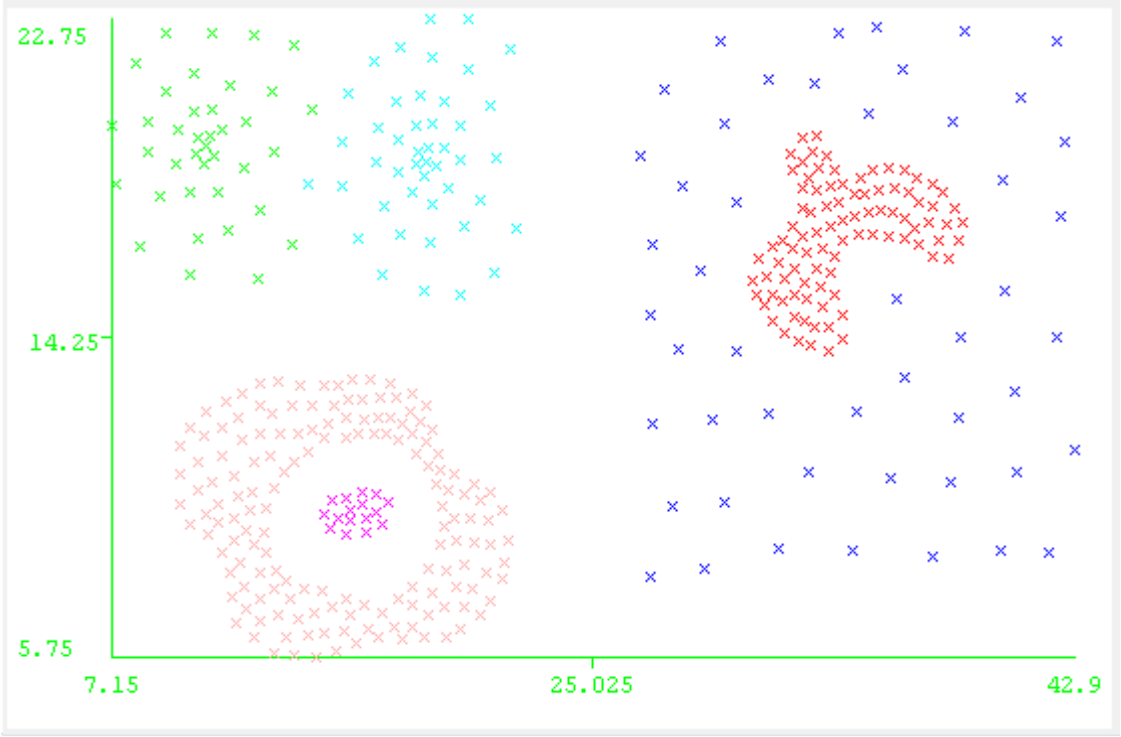
Considerazioni

Dalla seguente prova si evince il problema principale dell'approccio grid-based, ovvero che celle confinanti sono mergiate in un unico cluster potrebbero , al loro interno , contenere lo spazio di separazione tra i due cluster .
Difatti la problematica principale che ha portato ad una perdita di performance per entrambi gli algoritmi è dovuto proprio all'unione dei due cluster centrali in figura.

Test Compound

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 400
- Numero di cluster → 6 senza rumore

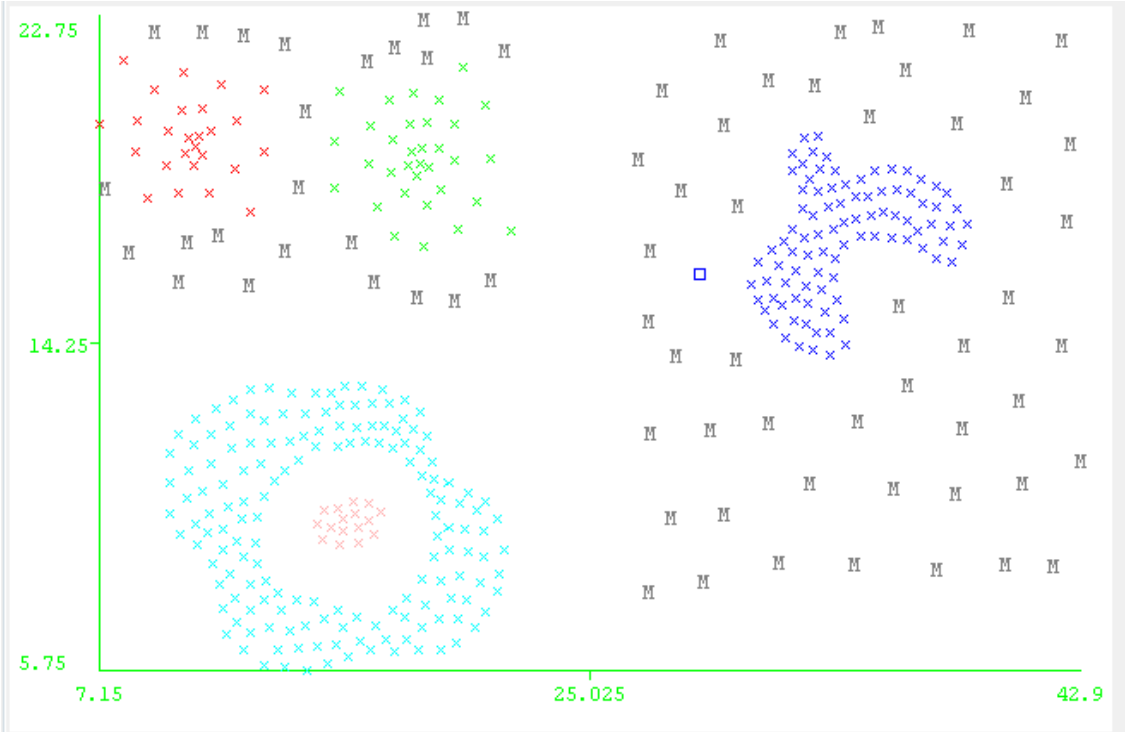


Risultati :

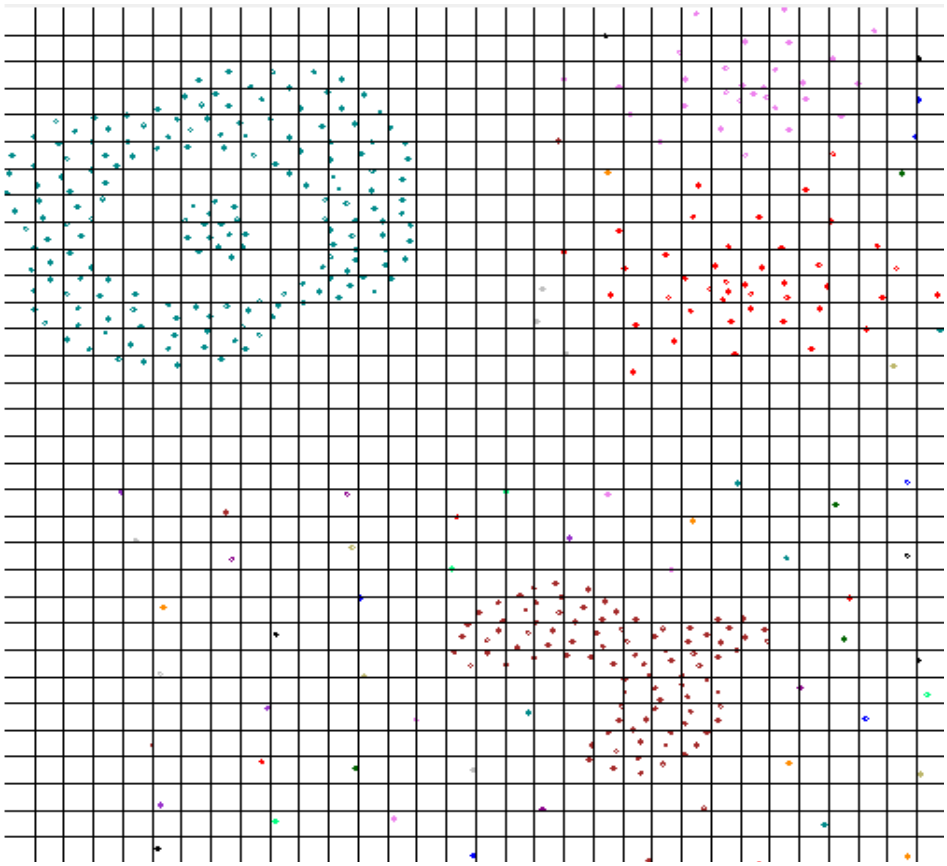
- Clustering statistico versione 2 → N=78 Confidenza =0.36
  - Numero di istanze errate → 78 ovvero il 20,15%
- Clustering statistico versione 3 → N=75 Confidenza =0.65
  - Numero di istanze errate → 75 ovvero il 18,7%
- DBSCAN E 0.056 MinPts 4
  - Numero di istanze errate → 73 ovvero il 18,15%

Risultati Grafici Degli algoritmi

DB scan



Clustering statistico 3.0 e 2.0 (simili)



Considerazioni

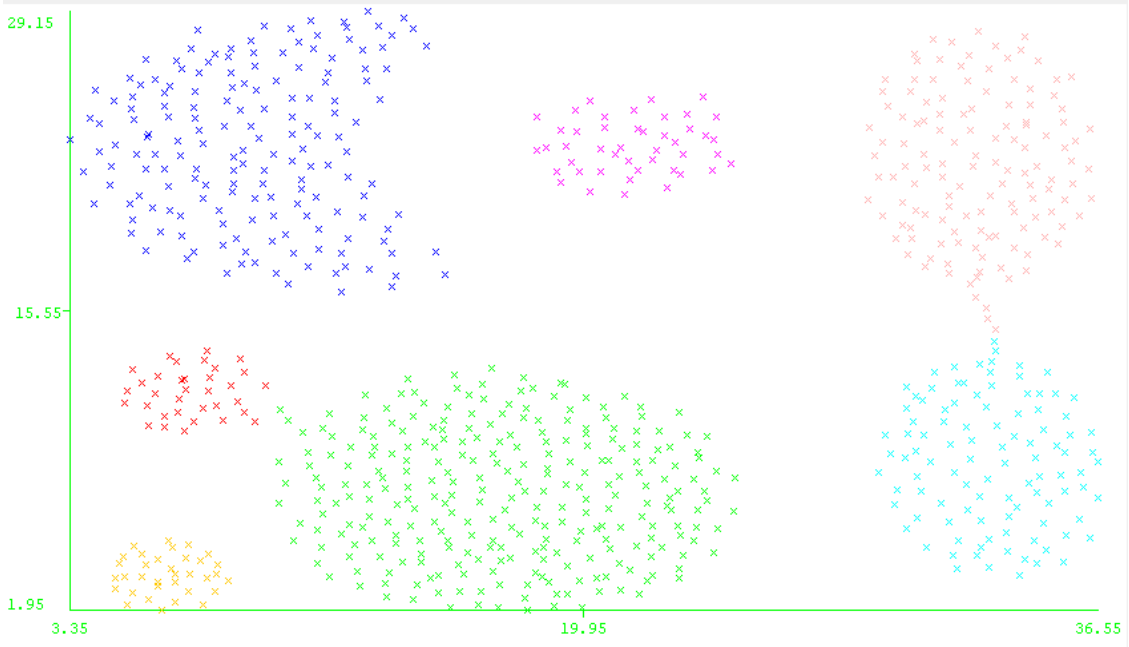
Nella prova effettuata è stato preso come confronto solo DBScan in quanto individuato come l'algoritmo migliore dei 3 di confronto.

In questo test i risultati ottenuti da tutti gli algoritmi sono pressochè simmetrici , ove la problematica principale è l'individuazione del cluster sparso colorato di blu , in quanto la propria densità risulta molto differente a quella degli altri cluster in esame.

Test Aggregation

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 788
- Numero di cluster → 6 senza rumore

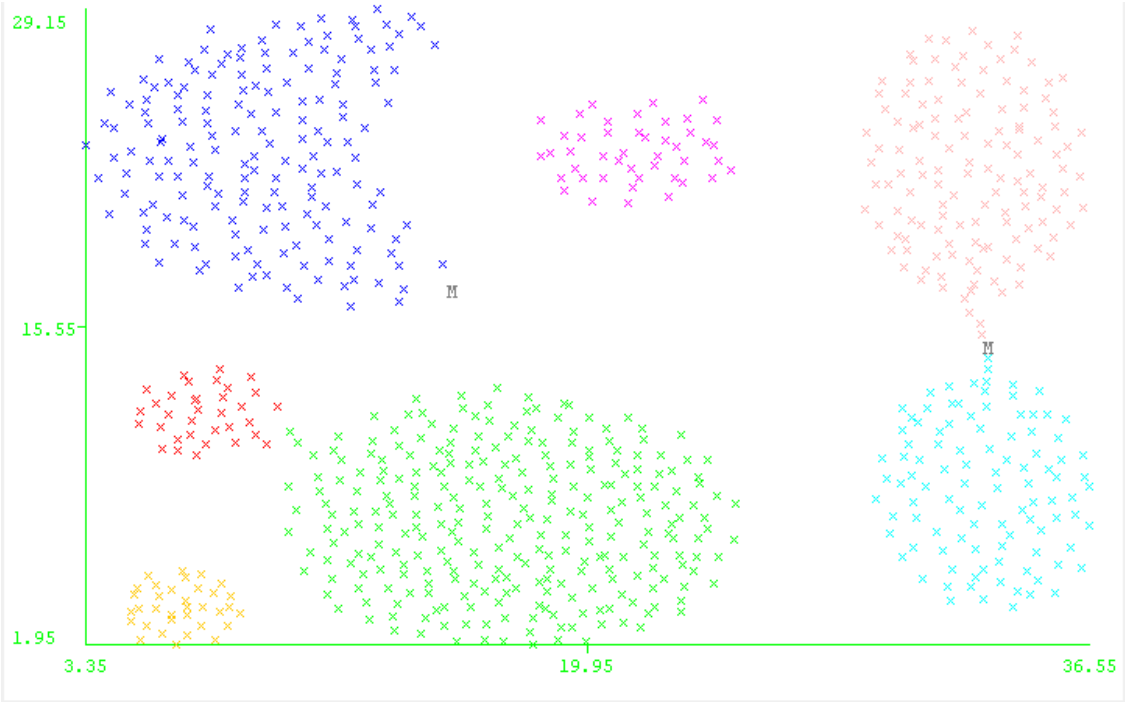


Risultati :

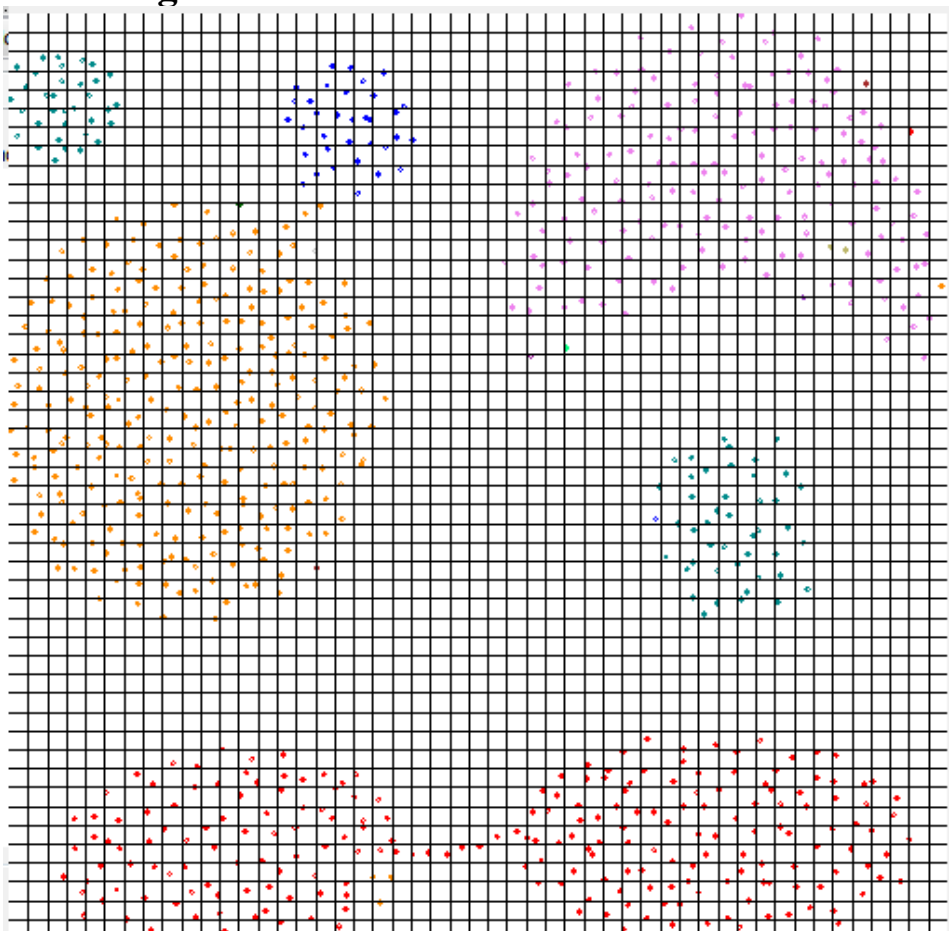
- Clustering statistico versione 2 → N=49 Confidenza =0.01
  - Numero di istanze errate → 41 ovvero il 5,30%
- Clustering statistico versione 3 → N=49 Confidenza =0.54
  - Numero di istanze errate → 141 ovvero il 17,89%
- DBSCAN E 0.056 MinPts 4
  - Numero di istanze errate → 2 ovvero il 0,25%

Risultati Grafici Degli algoritmi

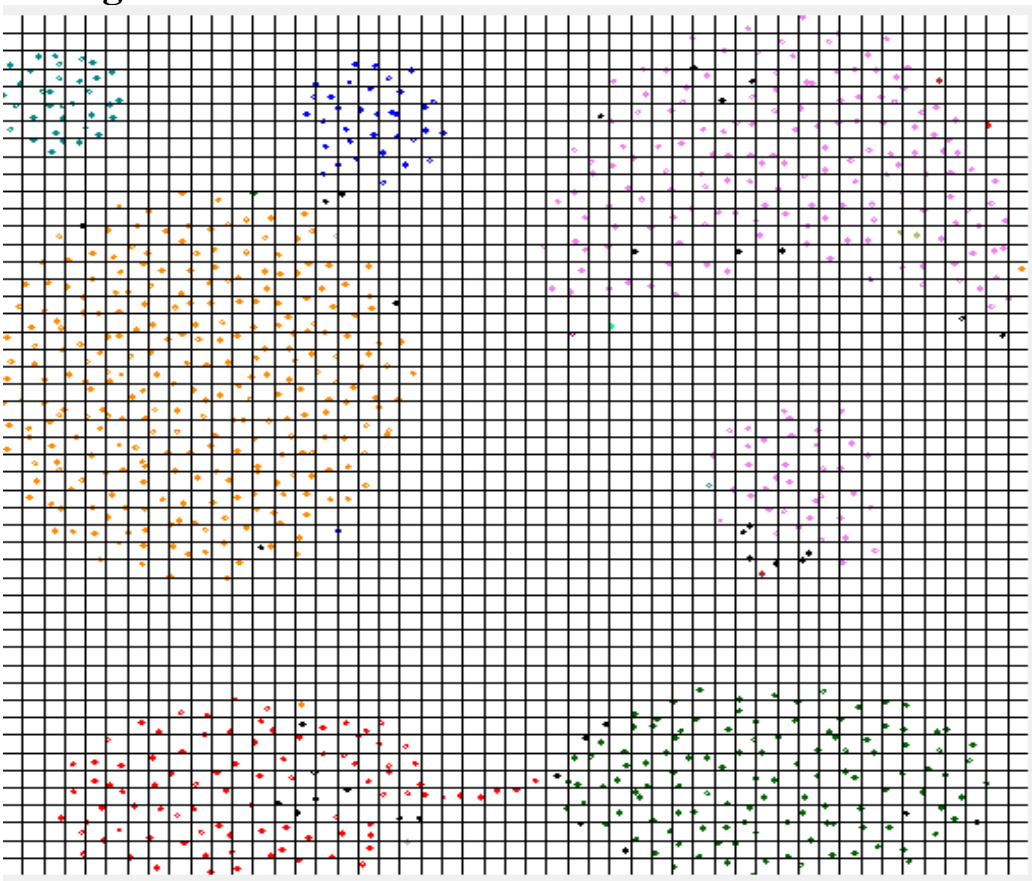
DB scan



Clustering statistico 3.0



Clustering statistico 3.0



Considerazioni

Nella prova effettuata è stato preso come confronto solo DBScan in quanto individuato come l'algoritmo migliore dei 3 di confronto.

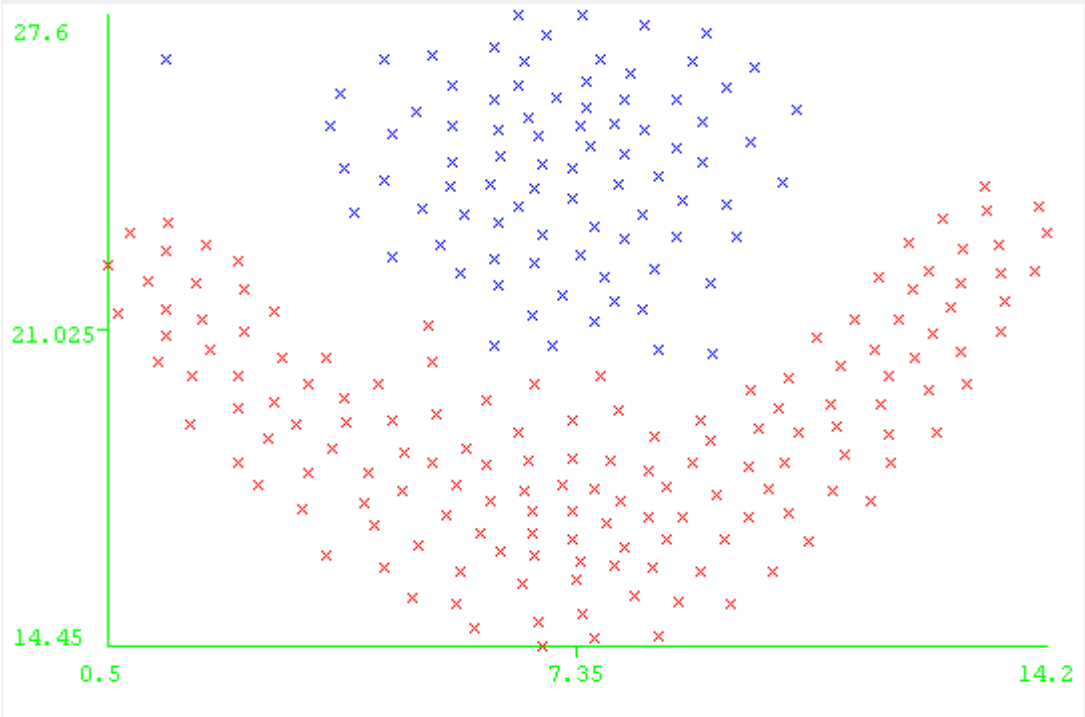
In questo test i risultati ottenuti sono migliori per DBScan , in quanto l'agglomerazione di cluster tramite percorsi possibili è un punto debole dell'algoritmo di cluestering utilizzato , difatti si trova “percorso denso + “ da seguire e di fatto alcuni cluster sono uniti tra di loro.

In questa prova la versione 2.0 risulta migliore della 3.0 in quanto la dinamica di densità / intorno gioca a favore della casualità quando vi è , come in questo caso , un percorso uniforme denso e stretto.

Test Flames

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 240
- Numero di cluster → 2 senza rumore

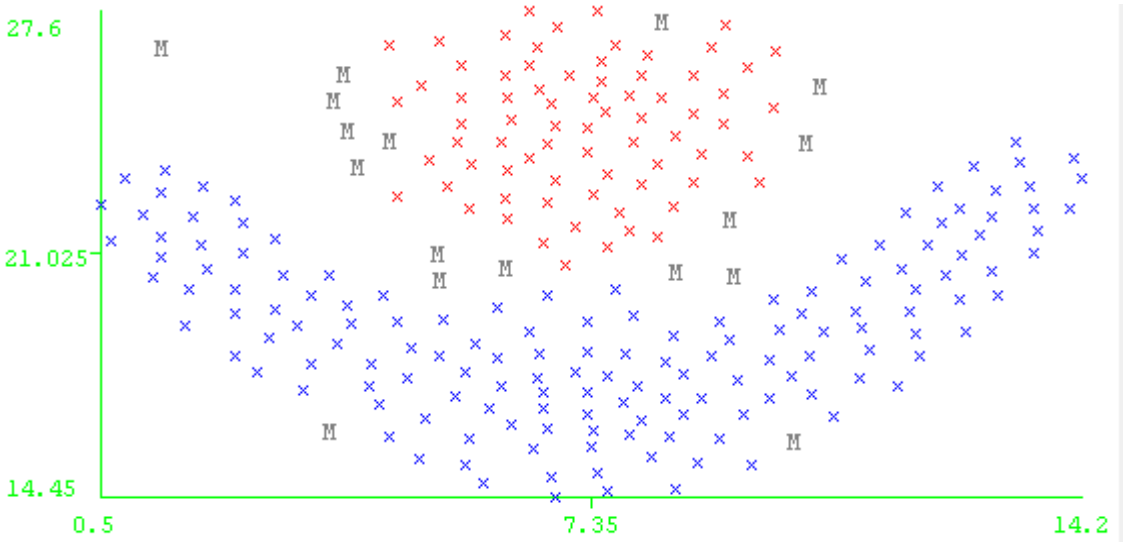


Risultati :

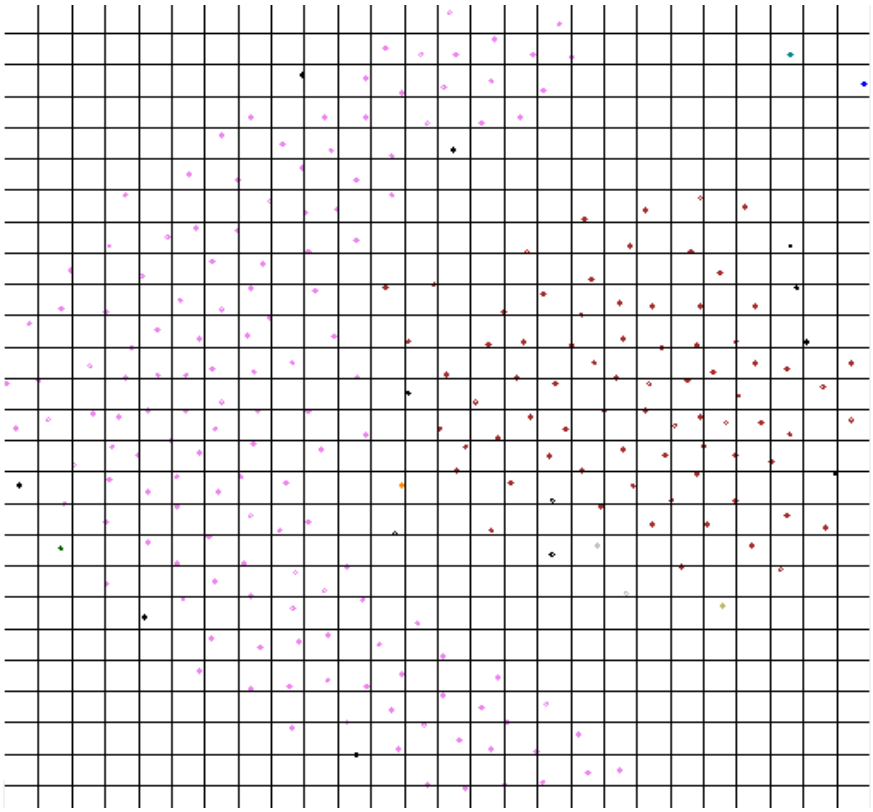
- Clustering statistico versione 2 → N=26 Confidenza =0.01
  - Numero di istanze errate → 18 ovvero il 7,5%
- Clustering statistico versione 3 → N=26 Confidenza =0.25
  - Numero di istanze errate → 12 ovvero il 5 %
- DBSCAN E 0.056 MinPts 4
  - Numero di istanze errate → 17 ovvero il 7,2%

Risultati Grafici Degli algoritmi

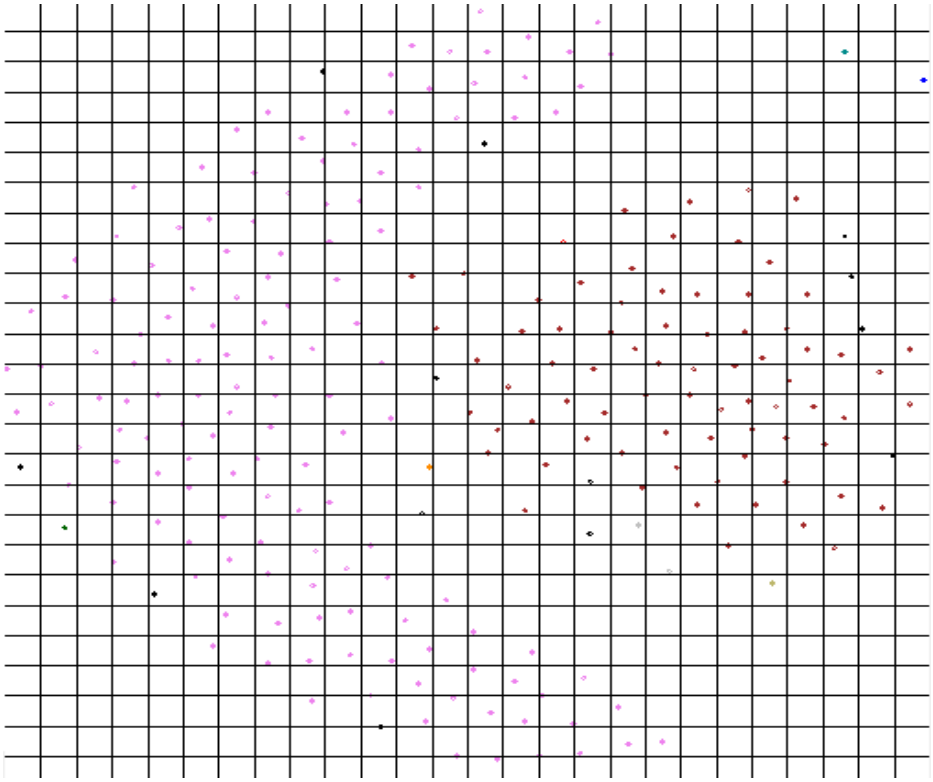
DB scan



Clustering statistico 2.0



Clustering statistico 3.0



Considerazioni

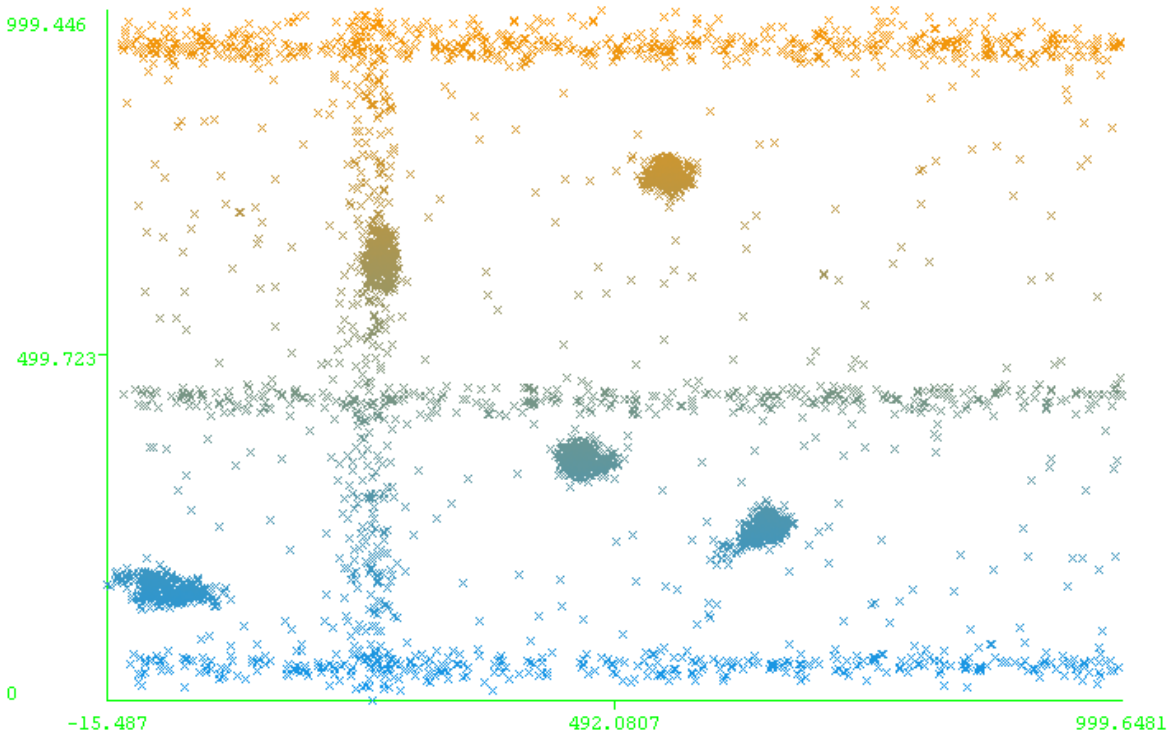
Nella prova effettuata è stato preso come confronto solo DBScan in quanto individuato come l'algoritmo migliore dei 3 di confronto.

In questa prova tutti e 3 gli algoritmi si equivalgono .

Test Line&Cluster.

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 3720
- Numero di cluster → 5 + rumore

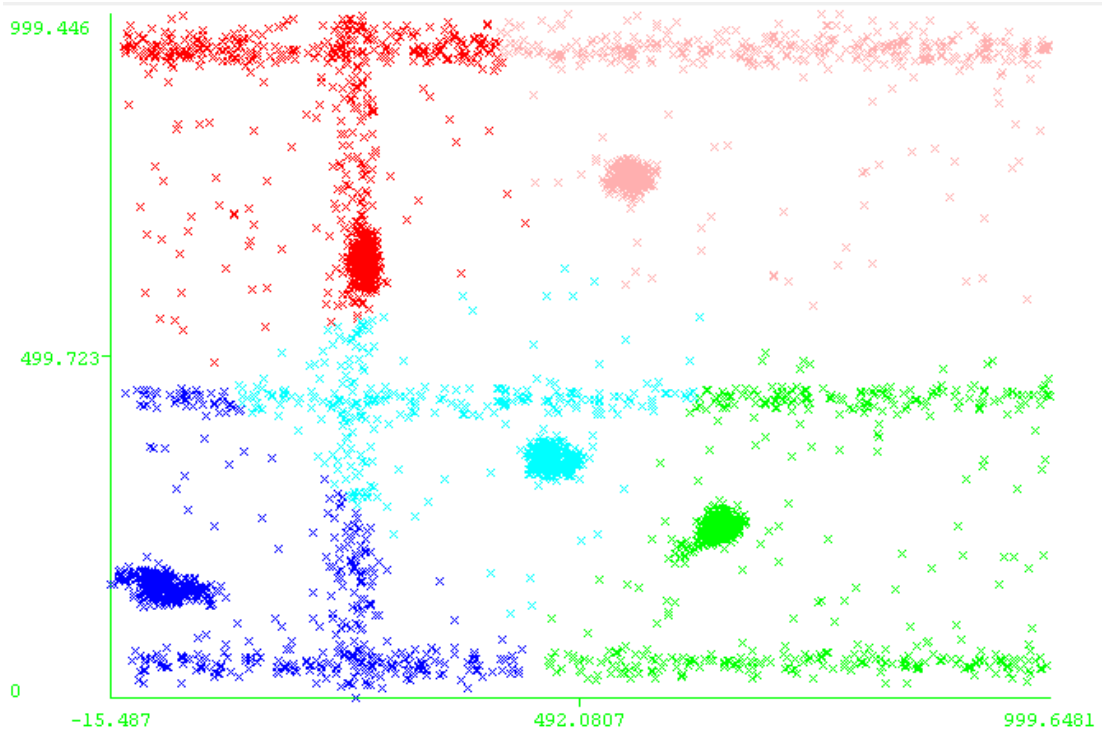


Risultati :

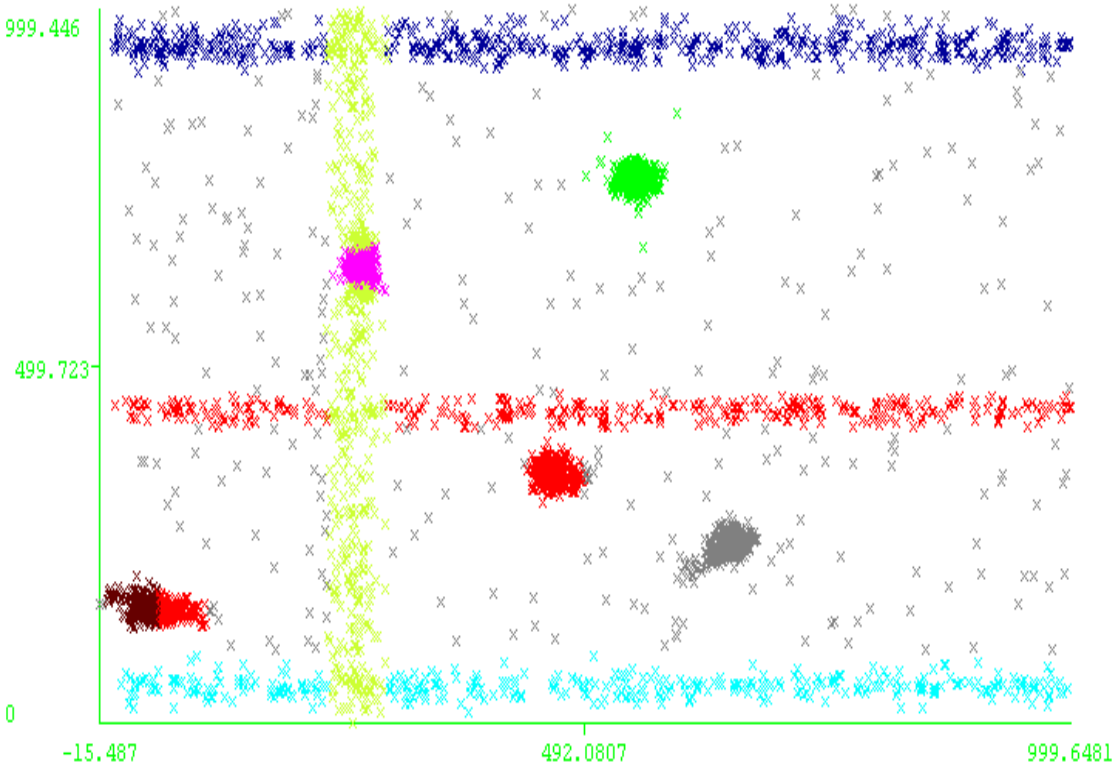
Per il seguente dataset non è stata fornita l'etichettatura, ma è stato comunque preso in considerazione per la sua particolare conformità, si riportano quindi i soli risultati grafici di per se già esplicativi del comportamento dei vari algoritmi di clustering.

Risultati Grafici Degli algoritmi

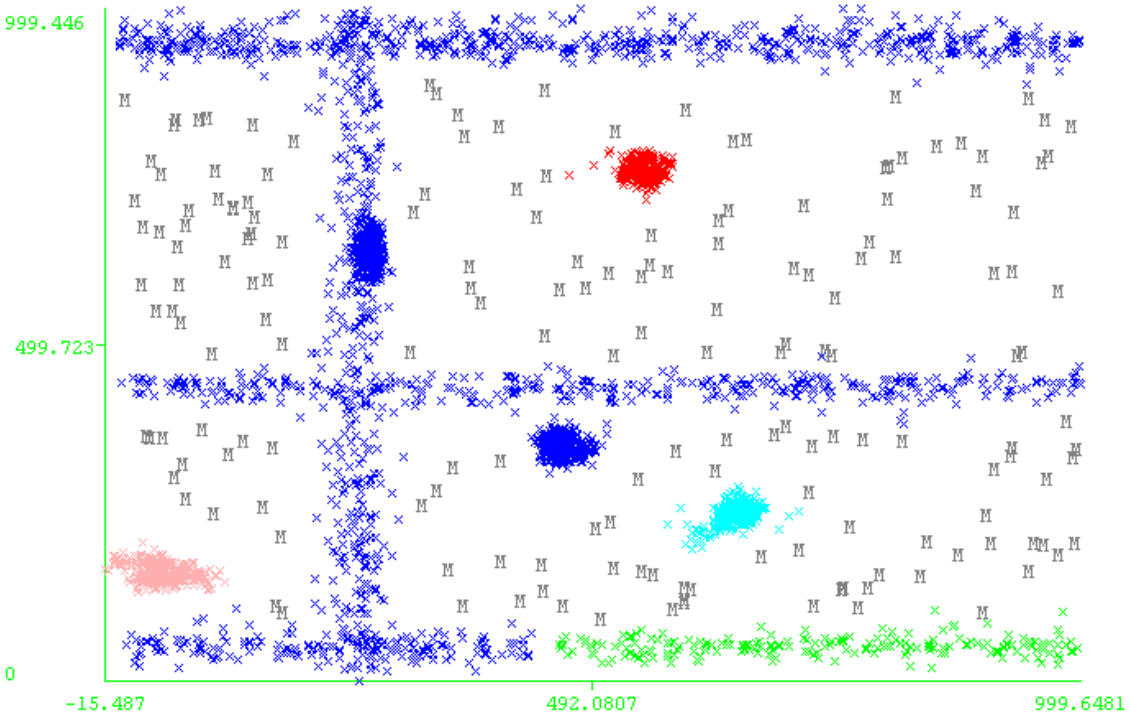
K-means



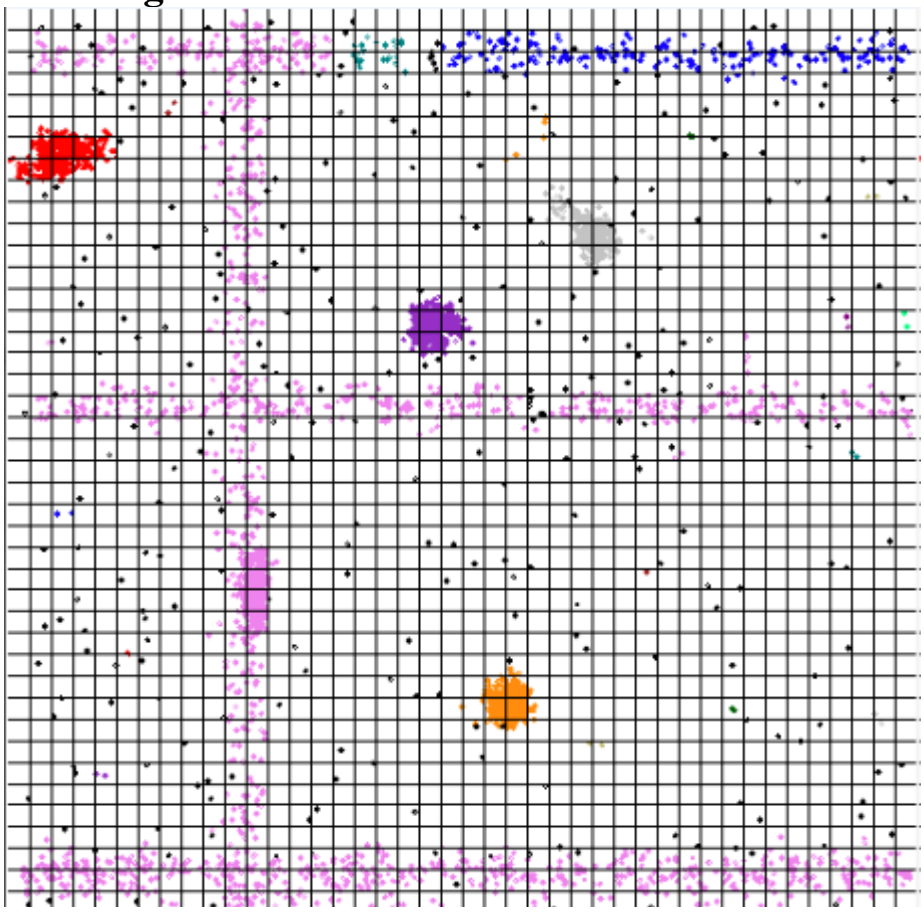
CobWeb



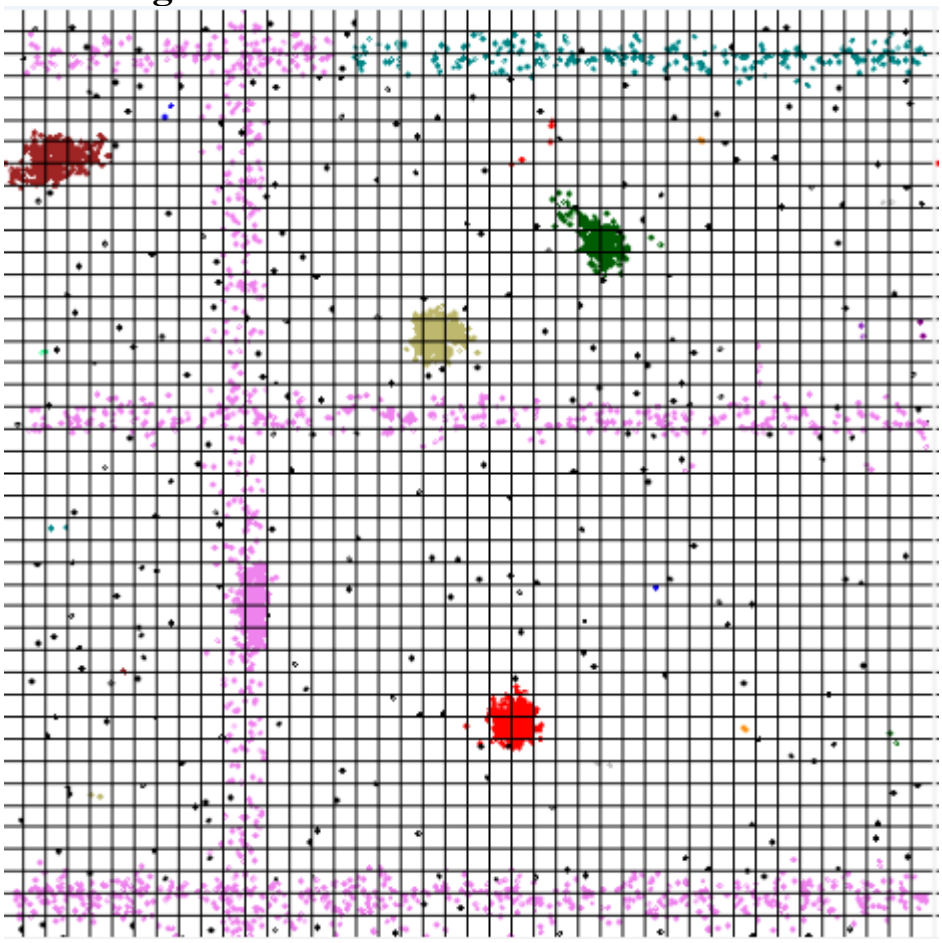
DB scan



ClusteringStatistico V 2.0



Clustering statistico 3.0



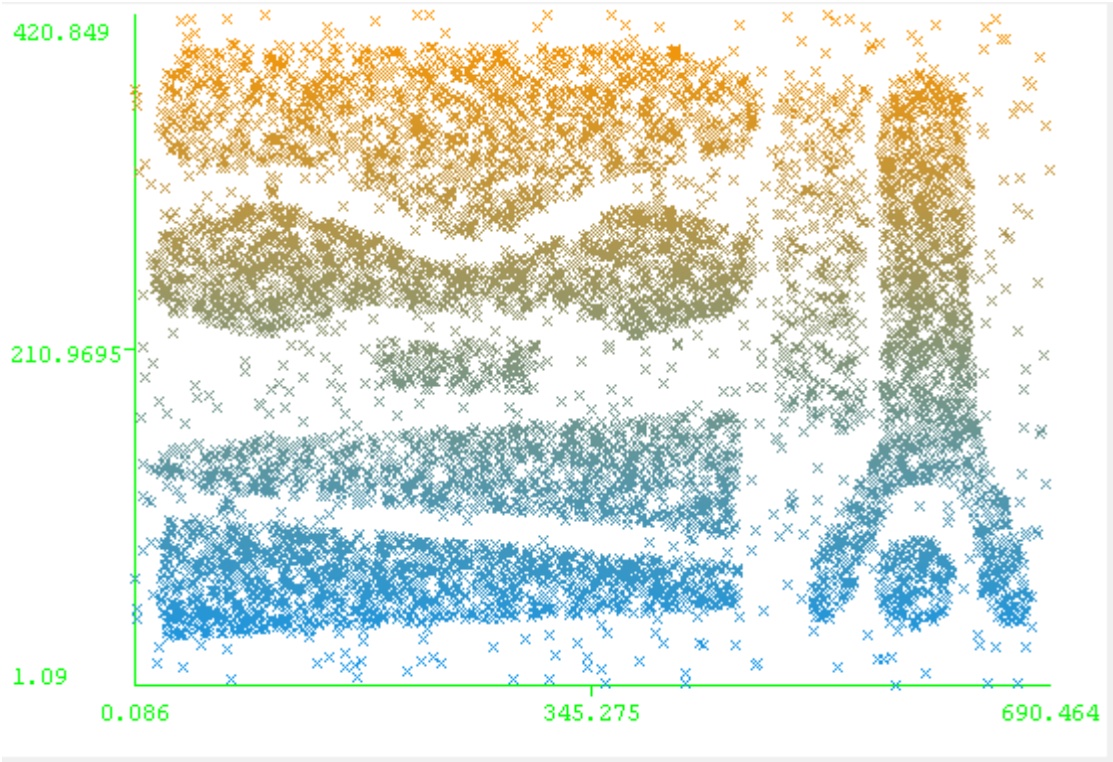
Considerazioni

L'algoritmo è resistente anche a diverse differenze di densità tra i cluster, si nota come le strisce verticali ed orizzontali siano a densità nettamente inferiore alla densità dei cluster centrali ; tale differenza se però raggiunge la critica soglia della media , per com'è impostato l'algoritmo di clustering attuale , crea una rottura all'interno del cluster.
Anche in questo caso la versione 3.0 performa leggermente meglio della versione 2.0 .

Test Dataset4.

Caratteristiche del Dataset :

- Numero di dimensioni → 2
- Numero di istanze → 8000
- Numero di cluster → 8 + rumore



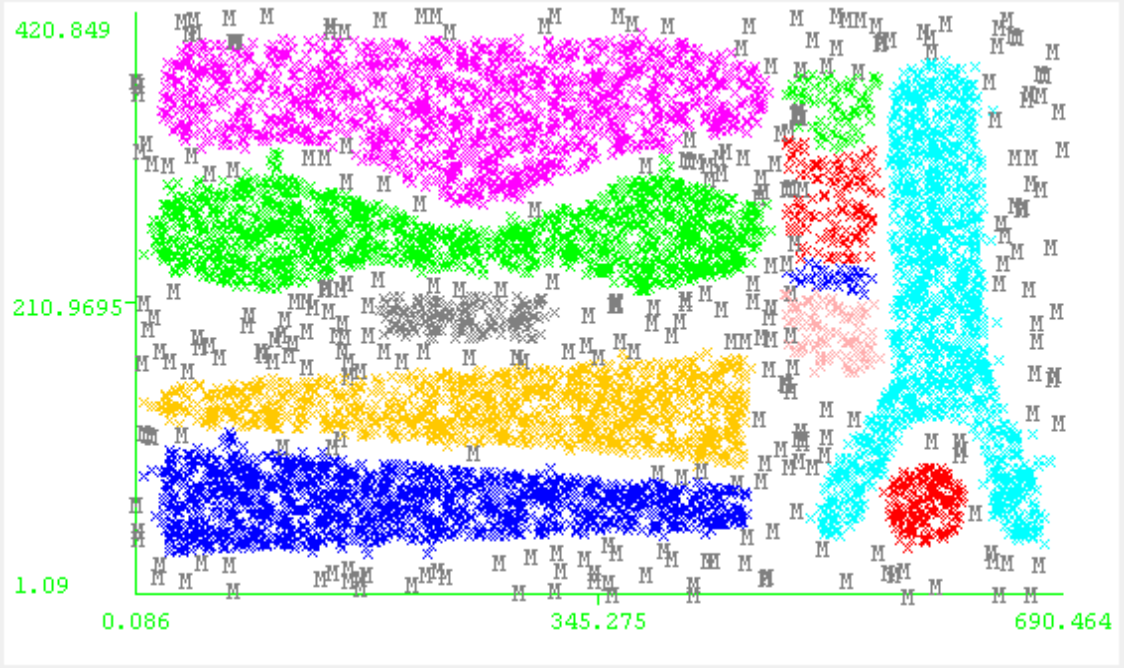
Risultati :

Per il seguente dataset non è stata fornita l'etichettatura, ma è stato comunque preso in considerazione per la sua particolare conformità, si riportano quindi i soli risultati grafici di per se già esplicativi del comportamento degli algoritmi significativi ovvero DBScan, Clustering statistico 2.0 e Clustering Statistico 3.0
Le prove effettuate hanno raggiunto l'ottimo tramite i parametri :

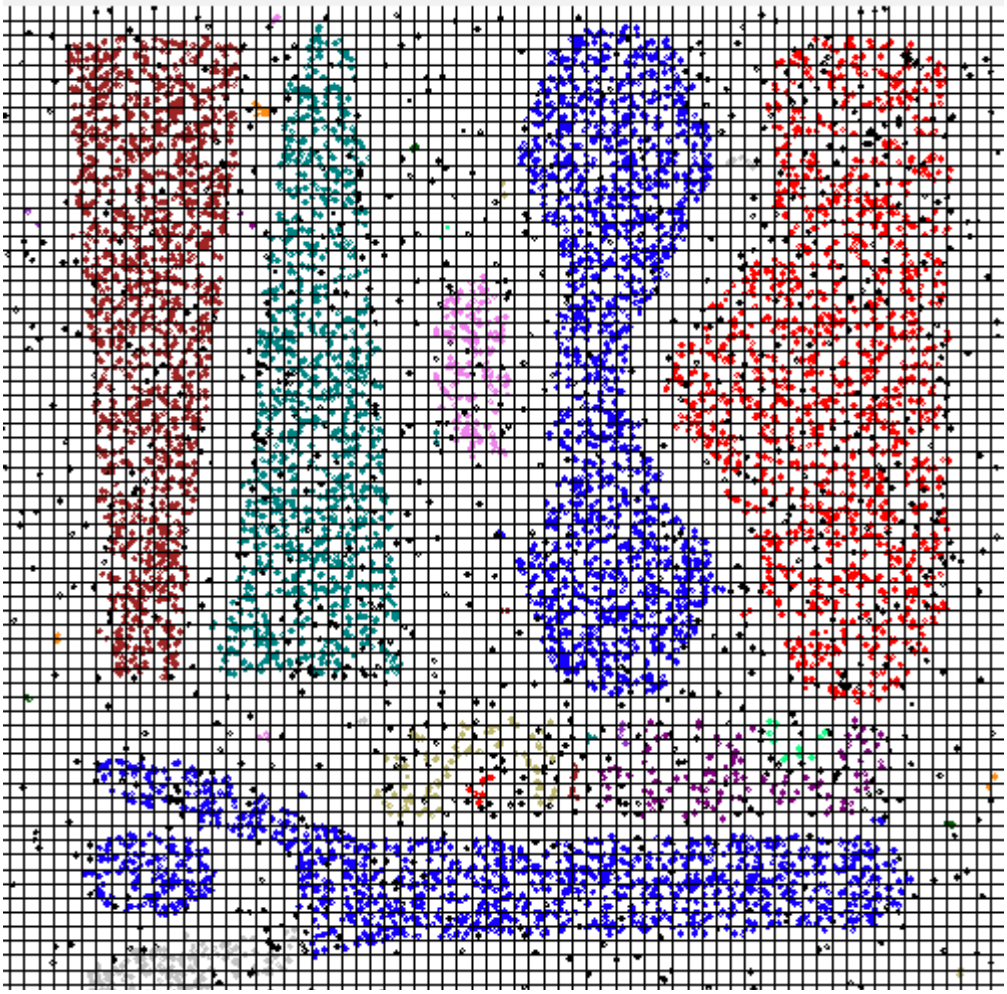
- Algoritmo 2.0 → N = 70 C = 0.35
- Algoritmo 3.0 → N = 49 C = 0.50
- DBScan → E= 0.018 MinPts = 4

Risultati Grafici Degli algoritmi

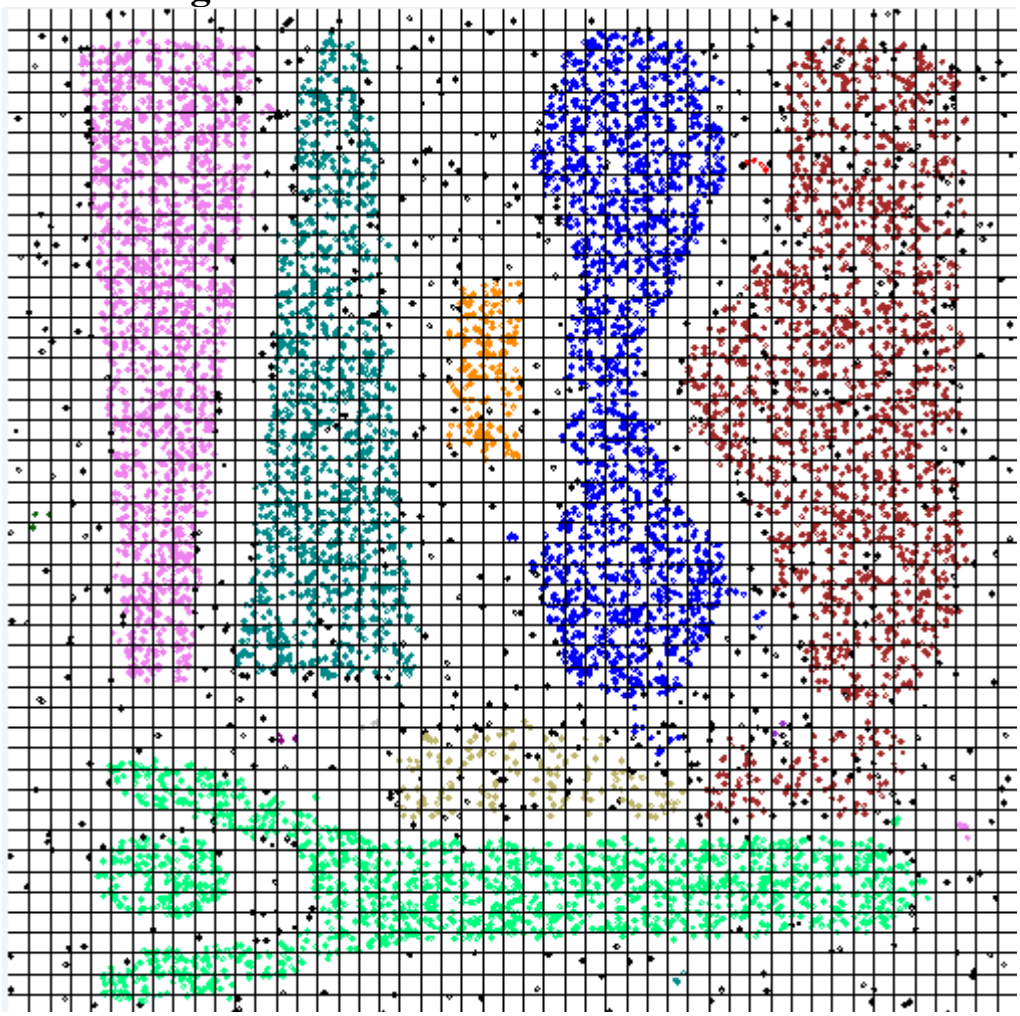
DB scan



ClusteringStatistico V 2.0



Clustering statistico 3.0



Considerazioni

Problematica comune per DBScan e gli algoritmi implementati è il riconoscimento della “differenza di densità” .
Ovvero si nota in figura come alcuni cluster siano più sparsi di altri. Questo porta alla frammentazione del meno denso in più cluster di dimensione inferiore.

Considerazioni finali

Dai risultati ottenuti si evidenziano sia punti di forza che punti deboli negli algoritmi di clustering statistico , nel dettaglio i punti a favore sono :

- Elevata resistenza al rumore
- Ottima capacità di individuazione di cluster di forme complesse
- Elevate performance anche in caso di dataset non rumorosi.

I punti a sfavore sono invece più attribuibili alle tecnologie “accessorie” all'algoritmo per l'individuazione delle regioni casuali e non casuali , ovvero l'utilizzo della griglia multi-dimensionale ed il semplice algoritmo di clustering implementato.
Si evince come un'ottimizzazione di questi due elmenti potrebbe portare tale algoritmo a risultati ancora migliori di quelli già ottenuti.

Algoritmo/Dataset	Dataset S	GirandBis Bis	Galaxy	Ex2m50	T4.8k	T7.10K
CluStat 2.0	13,01%	1,38%	6,55%	3,09%	15,15%	9,09%
CluStat 3.0	13,15%	1,38%	0,70%	3,09%	14,60%	8.90%
DBScan	28,69%	0,00%	0,10%	0,50%	3,15%	4.88%
CobWeb	67,19%	26,84%	14,67%	48.19%	68,49%	70,16%
K-means	60,16%	49,68%	20.17%	38,19%	34,21%	56,30%

Nella tabella sono riassunti i risultati ottenuti , si nota come DBScan risulti ottimo nel caso di dataset senza rumore, mentre quando il rumore aumenta tende ad aumentare l'errore sui cluster.
L'algoritmo di clustering statistico 3.0 è risultato leggermente superiore al 2.0 , anche se le differenze minime fanno presupporre che la condizione di indipendenza di intorno sia meno influente del previsto.
Complessivamente entrambi gli algoritmi risultano al pari di DBScan per datasets privi di rumore e migliore di quest'ultimo nei

dataset più rumorosi.

Conclusioni

In questo lavoro di tesi è stato dimostrato come è possibile cambiare l'approccio al clustering da "ricerca delle relazioni tra i dati " a "confronto tra il caso di assenza di relazioni ed il caso reale" . Un altro punto molto importante, che potrebbe significare molto anche in termini di studi di marketing, è che tale approccio evidenzia anche le aree DENSE – , ovvero le aree dove vi è una relazione negativa .
Ad esempio un indagine sulle aree DENSE- potrebbe portare un'azienda ad evitare settori di mercato che non porterebbero alcun beneficio , o un indagine sui propri clienti potrebbe far notare parti di mercato "scoperte".
L'approccio Grid-Based permette di abbattere i costi computazionali qual'ora il numero di regioni sia << del numero di punti del Dataset; tale tipologia di investigazione inoltre ha dimostrato un'ottima resistenza al rumore.
Essendo uno studio statistico però richiede che vi sia un'elevata numerosità dei dati all'interno delle regioni.
Dai risultati ottenuti l'approccio tiene testa agli algoritmi density-based che si sono rivelati i migliori in termini di performance , gli algoritmi di clustering basati su metodi statistici presentano però , per loro natura , una forte "resistenza" al rumore .
Si può quindi concludere che gli algoritmi di clustering statistico implementato offrono buone performance qualitative nel caso di dataset senza rumore al pari di un algoritmo density-based , mentre sono superiori nel caso di dataset più rumorosi , che più si avvicinano ai casi reali.

Sviluppi futuri.

I punti deboli evidenziati dagli algoritmi implementati sono diversi , di seguito saranno analizzati e da ognuno di essi potrà nascere un ulteriore studio che potrebbe portare ad un nuovo algoritmo di clustering statistico completo :

- Curse of Dimensionality : la maledizione della dimensionalità è il problema principale, in quanto l'approccio statistico è ottimo nel caso di regioni ad elevata numerosità ; ma con una griglia multidimensionale il numero di regioni aumenta esponenzialmente con il numero di dimensioni , stessa cosa per lo studio degli intorni di ogni regione ; gli interventi possibili sono 2 .
  - Effettuare suddivisioni intelligenti delle regioni , in base alle numerosità distinte di ogni regione
  - Applicare l'approccio statistico ad un algoritmo di Subspace-clustering, quindi identificare dimensione per dimensione le parti casuali / non casuali e poi applicare un meccanismo bottom-up di individuazione dei cluster
  - Definizione della griglia : l'approccio Grid based cambia l'ottimizzazione del risultato ottenuto in base ai tagli di ogni regione sullo spazio , trovare quindi l'ottimo numero di tagli per ogni dimensione diventa un task molto importante , e se fosse automatizzato risparmierebbe l'unico parametro "scomodo" dell'algoritmo ovvero N .
  - Clustering : l'algoritmo per il clustering dalle zone non casuali evidenziate presenta alcune lacune, in primis un semplice taglio sulla media della numerosità dei punti non è sufficente e porta ad errori, è necessario quindi investigare su migliori soluzioni.

Bibliografia

[1] H.Liu L.Parsons, E.Haque. Subspace clustering for high dimensional data:
A review. Sigkdd Explorations, 2004.

[2] I. T. Jollie. Principal Component Analysis. Springer, 2002.

[3] Singular value decomposition http://en.wikipedia.org/wiki/singular value decomposition.

[4] Sanjay Goil,Harsha Nagesh, Alok Choudhary. Maa: Ecient and scalable
subspace clustering for very large data sets.

[5] K.Kailing H.P. Kriegel P. Kroger. Density-connected subspace clustering for
high-dimensional data. 4th Siam International Conference on Data Mining
(pp.246-257), 2004.

[6] I.Assent R.Krieger E.Muller T.Seidl. Edsc: Efficient density based subspace
clustering. CIKM, 2008.

[7] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and
J. S. Park. Fast algorithms for projected clustering. In
Proceedings of the 1999 ACM SIGMOD international
conference on Management of data, pages 61{72. ACM
Press, 1999.

A Fast and Intelligent Subspace Clustering Algorithm using Dimension Voting, FINDIT [8]
[9]] R. Ng and J. Han. E±cient and e®ective clustering
methods for spatial data mining. In Proceedings of the
20th VLDB Conference, pages 144{155, 1994.

[10]] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In ACM SIGMOD, 2002

[11] M. L. Yiu and N. Mamoulis. Frequent-pattern based iterative projected clustering. In ICDM, pages 689{692, 2003

[12]Relevant Subspace Clustering: Mining the Most Interesting Non-Redundant Concepts in High Dimensional Data

[13]Subspace Clustering, Ensemble Clustering, Alternative Clustering, Multiview Clustering: What Can We Learn From Each Other?

[14]Scalable Density-Based Subspace Clustering

[15] GRIN

[16] GenIc: A Single Pass Generalized Incremental Algorithm for Clustering
Chetan Gupta*

[17] Evaluating Clustering in Subspace Projections of High Dimensional Data Emmanuel Muller ¨# Stephan Gunnemann ¨#Ira Assent # Thomas Seidl #

[18] Wikipedia: Data mining
http://en.wikipedia.org/wiki/data mining.

[19] M.Brescia. Review on clustering in data mining.

[20] M.Ester H.P.Kriegel J.Sander X.Xu. A density-based algorithm for
discovering clusters in large spatial databases with noise. 1996.

[21] Amit Banerjee Validating Clusters using; the HoDkins Statistic

[22] P. Holgatc, “Tcsts of randomness hascd on distance mcanucs;’ *Biometrika,*

[23] J. E. Besag and J. T. Glcaves, “On thc detection of spatial pattem in plant communitics,

[24] L. 1. Ebcrhardt. -'Some developments in distance sampling,”

[25]Anil K. Jain Data clustering: 50 years beyond K-means q

[26] http://it.wikipedia.org/wiki/Clustering

[27] http://it.wikipedia.org/wiki/Test_di_verifica_d%27ipotesi

[28] Chi-Square Test KL Kayser - The Annals of Thoracic Surgery, 1982 – Elsevier

[29] The Kolmogorov-Smirnov Test for Goodness of Fit a Frank J. Massey Jr.

[30] BR Page - Physics Teacher, 1995 The"Student" of the Student's t-Test

[31]Wilcoxon-Mann-Whitney test V DePuy, VW Berger, YY Zhou 2005

[32] http://www.mathematik.uni-kl.de/~schwaar/Exercises/Tabellen/table_kolmogorov.pdf

[33]J. B. MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations", Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297

[34] Fisher, Douglas (1987). "Knowledge acquisition via

incremental conceptual clustering"

[35] W. M. Rand (1971). "Objective criteria for the evaluation of clustering methods"