r t Anno 2011_Numero 182

rapporti tecnici INGV

PyDBSCAN un software per il clustering di dati

182

Istituto Nazionale di Geofisica e Vulcanologia

Segreteria di Redazione
Francesca Di Stefano - coordinatore
Tel. +39 06 51860068
Fax +39 06 36915617
Rossella Celi
Tel. +39 06 51860055
Fax +39 06 36915617

redazionecen@ingv.it

Anno 2011_Numero 182

rapporti tecnici

INGV

PYDBSCAN UN SOFTWARE PER IL CLUSTERING DI DATI

Carmelo Cassisi[2], Placido Montalto[1], Alfredo Pulvirenti[2], Marco Aliotta[1,2], Andrea Cannata[1]

[1]**INGV** (Istituto Nazionale di Geofisica e Vulcanologia, Sezione di Catania)
[2]**UNIVERSITÀ DEGLI STUDI DI CATANIA** (Dipartimento di Matematica e Informatica)

182

Indice

Introduzione 5

1. Metodi di clustering basati su densità 6

1.1 Stratificazione dello spazio e definizione degli outlier 7

1.2 Clustering di dataset con diverse densità di punti 11

2. Implementazione dell'algoritmo DBSCAN 11

3. Risultati ottenuti con il software PyDBSCAN 15

Conclusioni 17

Bibliografia 18

Introduzione

Con il termine *clustering* si indica il processo mediante il quale è possibile raggruppare oggetti in base a caratteristiche comuni (*features*). Questo approccio, alla base dei processi di estrazione di conoscenza da insiemi di dati (*data mining*), riveste notevole importanza nelle tecniche di analisi. Come verrà mostrato in questo lavoro, l'applicazione delle tecniche di clustering consente di analizzare dataset, con l'obiettivo di ricercare strutture che possano fornire informazioni utili circa i dati oggetto dello studio. Gli ambiti in cui tali algoritmi sono impiegati risultano essere eterogenei, a partire dalle analisi di dati biomedici, astrofisici, biologici, fino ad arrivare a quelli geofisici. La letteratura è ricca di vari casi di studio, dai quali il ricercatore può trarre spunto e adattare i differenti approcci alle proprie esigenze.

Il software *PyDBSCAN*, oggetto del presente lavoro, permette di applicare tecniche di clustering basate sul concetto di densità, applicate ad oggetti (o punti) appartenenti ad insiemi definiti in uno spazio metrico. L'algoritmo di base è il *DBSCAN* (*Density Based Spatial Clustering on Application with Noise*) [Ester et al., 1996], di cui viene riportata una implementazione ottimizzata al fine di migliorare la qualità del processamento dei dati. Schematicamente, il sistema proposto può essere rappresentato come in Fig. 1. Il software, sviluppato in Python 2.6 [Python ref.], utilizza le librerie scientifiche Numpy [Numpy ref.], Matplotlib [matplotlib ref.] e la libreria grafica PyQt [PyQt ref.] impiegata nella realizzazione dell'interfaccia utente. Python è un linguaggio di programmazione che permette la realizzazione di applicazioni cross-platform in grado di funzionare su diversi sistemi operativi quali Windows, Unix, Linux e Mac OS.

Nella prima parte del lavoro verranno brevemente descritte le tecniche oggetto del software presentato, mentre nella seconda parte verrà descritto un esempio di applicazione su dati reali.

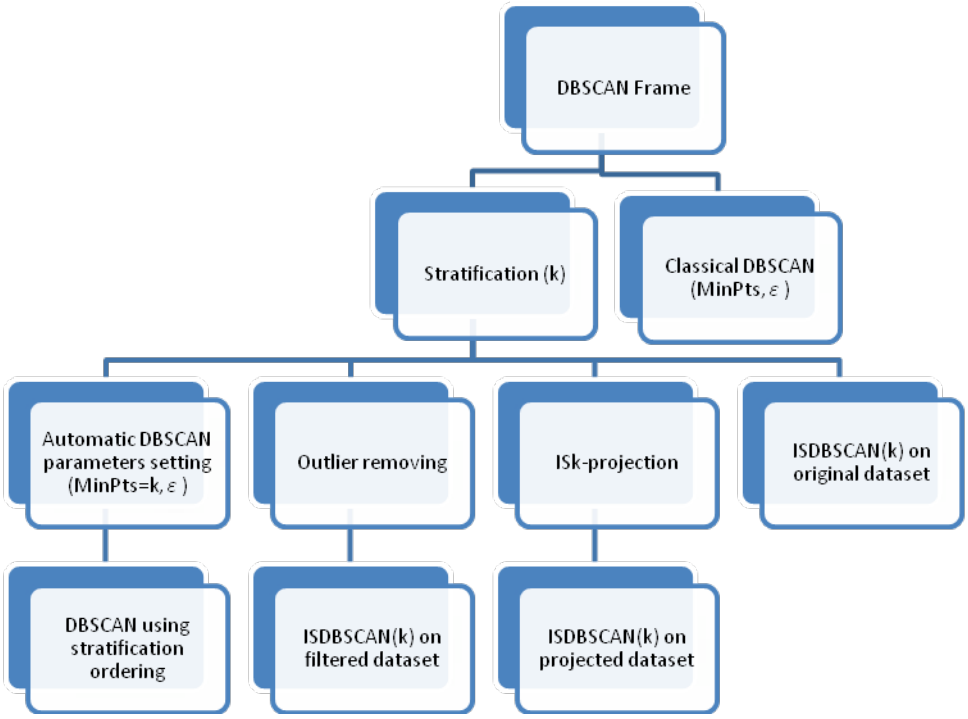


Figura 1. Schema del sistema proposto.

1. Metodi di clustering basati su densità

L'algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [Ester et al., 1996] è basato sull'idea che oggetti che formano regioni dense possono essere raggruppati in cluster. Gli oggetti sono *punti* in uno spazio *d*-dimensionale (R^d) nel quale viene definita una funzione di distanza tra due punti *p*, *q*: *dist*(*p*,*q*). Per velocizzare il calcolo delle distanze, l'algoritmo utilizza strutture di indicizzazione come *R** tree [Beckmann et al., 1990], *Antipole* tree [Cantone et al., 2005] o *kd* tree [Bentley, 1975]. DBSCAN accetta in input un dataset *D* e due parametri, ε e *MinPts*, per il calcolo delle densità.

Viene definito *ε-neighbourhood* di un punto *p* l'insieme di punti N_ε che ricadono nel cerchio di raggio ε e centro *p*. Se $|N_\varepsilon| \geq MinPts$ allora *p* viene chiamato *core point*. Tutti i punti in N_ε sono *direttamente raggiungibili per densità* (*Directly density-reachable*) da *p* (Fig. 2a). Un punto *q* è *densamente raggiungibile* (*Density-reachable*) da un punto *p* se esiste una catena di punti $q_1,...,q_n$, $q_1 = p$, $q_n = q$ tale che per ogni *i*, q_{i+1} è *direttamente raggiungibile per densità* da q_i, per $1 \leq i \leq n$ (Fig. 2b). Un punto *p* è *densamente connesso* (*density-connected*) a un punto *q,* se esiste un punto *o* tale che sia *p* che *q* sono *densamente raggiungibili* da *o* (Fig. 2c). Un cluster è un insieme massimale di punti *density-connected*.

L'algoritmo DBSCAN opera come segue: sceglie casualmente un punto *p* in *D* e controlla il suo *ε-neighbourhood* N_ε. Se N_ε contiene più di *MinPts*, crea un nuovo cluster con *p* come *core point*, e iterativamente aggiunge tutti i punti *direttamente raggiungibili per densità* da *p*.

Il processo termina quando non ci sono più punti da aggiungere al cluster. Verrà poi scelto in maniera casuale un nuovo punto non classificato e i passi precedentemente descritti verranno re-iterati finché non ci saranno più punti da assegnare a nessun cluster. Un punto in *D* è definito *outlier* se non è possibile assegnarlo a nessun cluster.

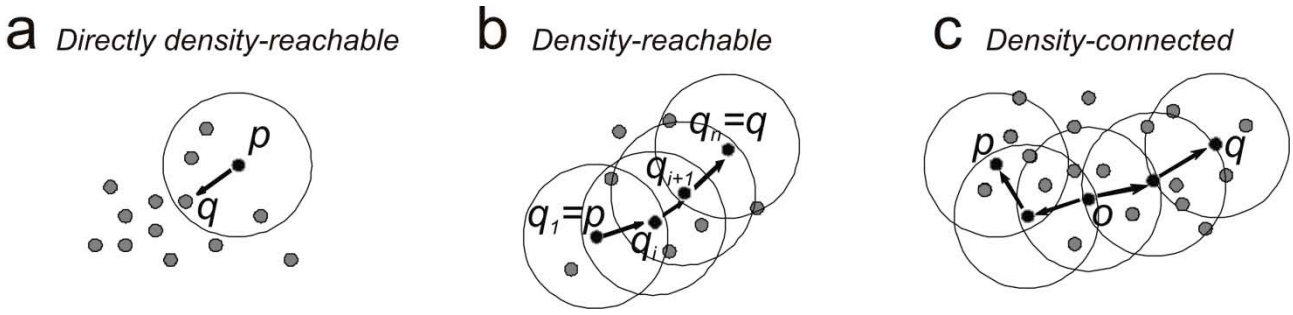
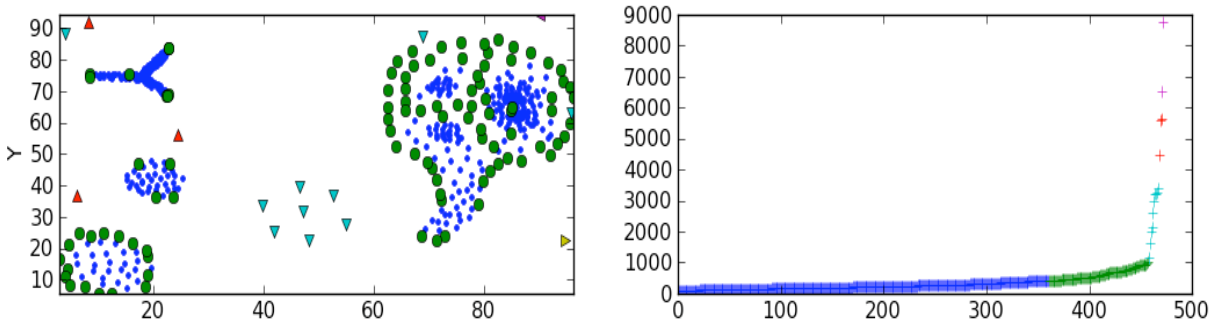


Figura 2. Esempi di (a) *directly density-reachable*, (b) *density-reachable*, (c) *density-connected* nel clustering basato su densità. Si supponga *MinPts = 3*. I punti neri e grigi indicano i punti da clusterizzare, i cerchi delineano l'area di raggio ε intorno ai punti neri e le frecce denotano la relazione di *directly density-reachability*. In (a) il punto *p* è *core point*, mentre *q* è *directly density-reachable* da *p*. In (b) il punto *q* è *density-reachable* da *p*. In (c) il punto *q* è *density-connected* a *p* e *o* è un punto tale che sia *p* che *q* sono *density-reachable* da *o*.

1.1 Stratificazione dello spazio e definizione degli outlier

La stratificazione può essere considerata come una fase di *pre-processing* del dato allo scopo di analizzare le sue proprietà spaziali principali. Essa dispone i punti di un dataset D su strati, dove punti appartenenti allo stesso strato hanno caratteristiche globali simili (Fig. 3). Scelta una particolare funzione f, che associa ad ogni punto p un valore di densità, essa costruisce un ordinamento tra i punti utilizzando il valore di $f(p)$ come chiave.

Se la funzione f viene scelta in maniera opportuna, le informazioni restituite dalla stratificazione risultano utili per migliorare le performance dell'algoritmo DBSCAN. In questo caso, invece di effettuare una selezione casuale del punto di partenza, si può iniziare l'espansione dei cluster dal punto più in basso nell'ordinamento, il quale risulterà avere le caratteristiche di un *core point* [Fahim et al., 2009] (Fig. 3).



(a) **(b)**

Figura 3. Punti colorati in maniera uguale rientrano nello stesso strato. Punti colorati in maniera differente a strati differenti. (a) Plot di un dataset di punti suddiviso in strati. (b) Ordinamento crescente dei punti in base al valore di $f(p)$.

Per proseguire nella trattazione dell'algoritmo occorre definire la funzione DF_k a partire dalle seguenti definizioni:

Definizione1. *k-distance e k-nearest neighbourhood of p.* La k-distance di p, $k_{dist}(p)$, è la distanza $dist(p,q)$ tra p e q in D, tale che: (1) per almeno k punti $q' \in D$ risulta $dist(p,q') \leq dist(p,q)$ e, (2) al massimo per $(k-1)$ punti $q' \in D$ risulta $dist(p,q') < dist(p,q)$. Il *k nearest neighbourhood* di p, $NN_k(p)$, è un insieme di punti X in D con $dist(p,X) \leq k_{dist}(p)$: $NN_k(p) = \{X \in D \setminus \{p\} \mid dist(p,X) \leq k_{dist}(p)\}$

Definizione 2. Somma delle prime k distanze dei punti in $NN_k(p)$: $\omega_k(p)$. Sia $NN_k(p)$ l'insieme dei k nearest neighbours di p, la somma delle distanze di ogni suo punto da p è indicata come: $\omega_k(p) = \sum_{q \in NN_k(p)} dist(p,q)$.

Definizione 3. *Reverse nearest neighbourhood di p*: la relazione di vicinato inversa è definita come: $RNN_k(p) = \{q \mid q \in D, p \in NN_k(q)\}$.

Definizione 4. *Influence space* di p,$IS_k(p)$. La *k-Influence Space* di p è definita come: $IS_k(p) = RNN_k(p) \cap NN_k(p)$

Definizione 5. Densità locale di p. La densità di p, $den(p)$, è l'inversa della k-distance (Def. 1) di p : $den(p) = 1/k_{dist}(p)$.

Definizione 6. *AINFLO di p*: Sia $INFLO_k(p)$ l'indice di outlierness calcolato come: $INFLO_k(p) = \frac{den_{avg}(IS_k(p))}{den(p)}$, dove $den_{avg}(p) = \frac{\sum_{o \in IS_k(p)} den(o)}{|IS_k(p)|}$ [Jin et al., 2006]. L' *AINFLO* (*Adjusted INFLuenced Outlierness*) di *p* è definito come: $AINFLO_k(p) = \frac{INFLO_k(p)}{|RNN_k(p)|}$.

Definizione 7. *Density Function $DF_k(p)$*. Normalizzando il valore di $AINFLO_k$(p) a ω_k(p), e combinando linearmente questi due valori, si ottiene la seguente funzione di densità: $DF_k(p) = AINFLO_k(p) + \omega_k(p)$.

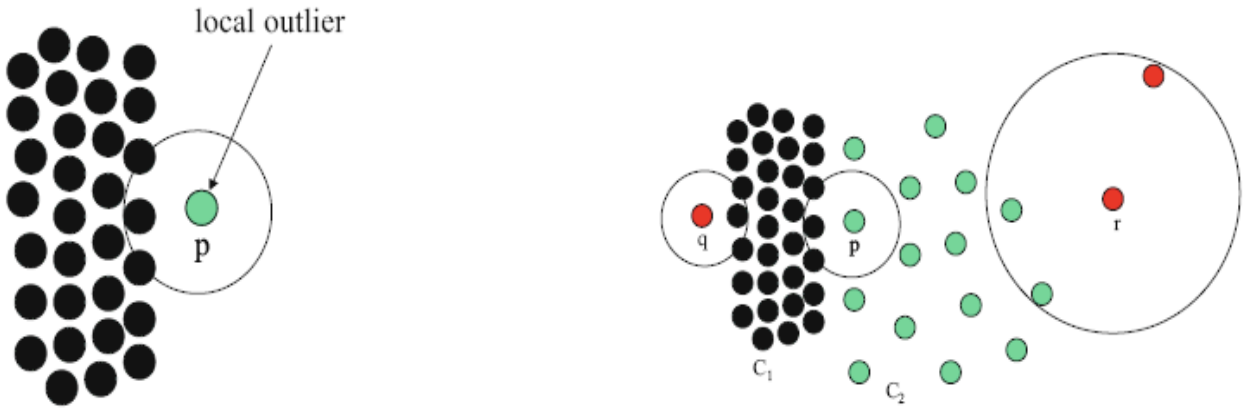


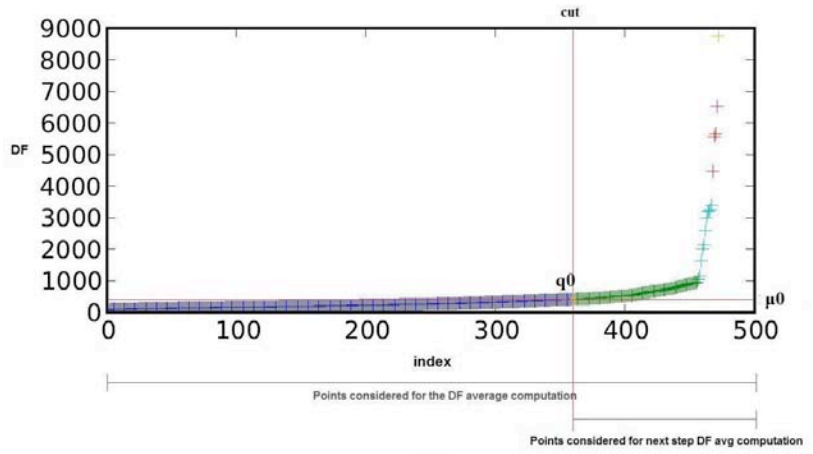
Figura 4. Sulla sinistra, il punto *p* può essere considerato come un outlier locale. Sulla destra, p non può essere considerato outlier come *q*, perché appartenente ad una regione di punti meno densa. [Immagine da Jin et al., 2006].

La funzione DF_k sopra definita restituisce per un dato punto *p* il valore della somma delle prime *k* distanze dei punti in $NN_k(p)$ e un valore di *outlierness* derivato dalla relazione proposta in Jin et al. (2006), capace di discriminare sia gli outlier locali che quelli globali (Fig. 4).

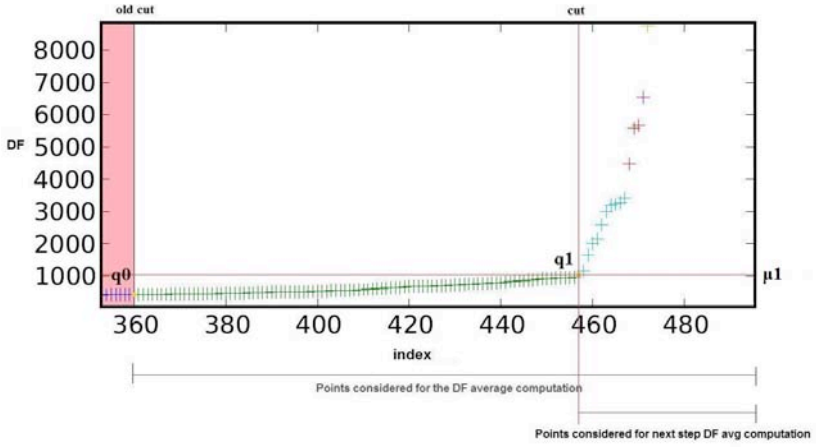
Un esempio del processo di partizionamento del dataset, riportato in Fig. 5 è di seguito descritto. Inizialmente si parte dalla curva dei valori DF_k ordinati in maniera crescente. Definito il valore medio di tutti i valori DF_k con $\mu_0 = \frac{\sum_{p \in D} DF_k(p)}{|D|}$, la prima suddivisione viene effettuata assegnando al primo strato tutti i punti nell'ordinamento che stanno nelle posizioni precedenti al punto q_0 avente $f(q_0)=\mu_0$. Si reitera dunque lo stesso processo assegnando, al passo *i*-esimo, tutti i punti nell'ordinamento compresi tra q_{i-1} e q_i, dove $f(q_i)=\mu_i$ è il valore medio calcolato sui punti del dataset non ancora inseriti in nessuno strato al passo *i*-esimo. La procedura si ferma quando il valore di *outlierness* (*AINFLO*) medio dei punti considerati allo step *i*-esimo non è più alto del valore di *outlierness* medio del dataset.

Con questa tecnica, i punti più outlier vengono disposti negli ultimi strati dell'ordinamento (Fig. 3). Viene inserita inoltre una procedura di estrazione degli outlier, che restituisce una più accurata bipartizione tra punti interni e punti outlier (Fig. 6b,c), poiché analisi svolte su vari dataset hanno mostrato che gli outlier non si dispongono sempre e solo nell'ultimo strato dell'ordinamento.

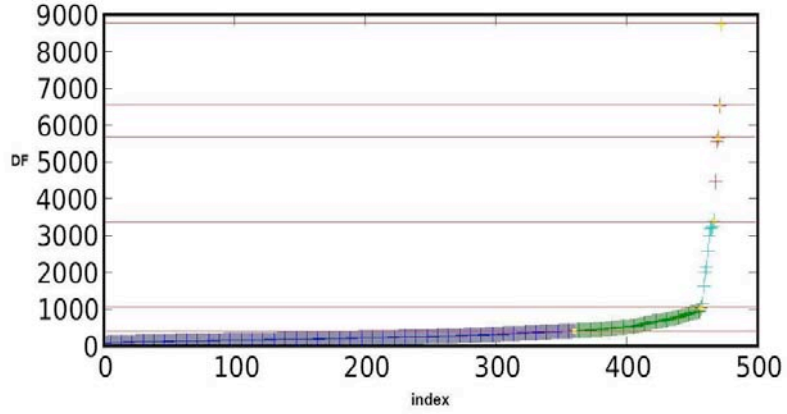
Viene quindi effettuata una correzione sulla divisione tra i possibili outlier e i punti potenzialmente appartenenti ai cluster, utilizzando una soglia δ calcolata come media tra la pendenza della retta che approssima i valori dei punti del primo strato e quella della retta relativa all'ultimo strato (Fig. 6a). La procedura inizia dal primo indice *i* del secondo strato (valori verdi in Fig.6a), calcolando il valore $DF(i) + DF(i+1) / 2$, e si ferma non appena trova una coppia di valori consecutivi ($DF(j)$, $DF(j+1)$), con $j > i$, tali che $DF(j) + DF(j+1) / 2 > \delta$. L'indice *j* verrà considerato come *S.O.I.* (*Start Outlier Index*), cioè come primo punto outlier dell'ordinamento.



(1)



(2)



(final)

Figura 5. Passi del processo di stratificazione.

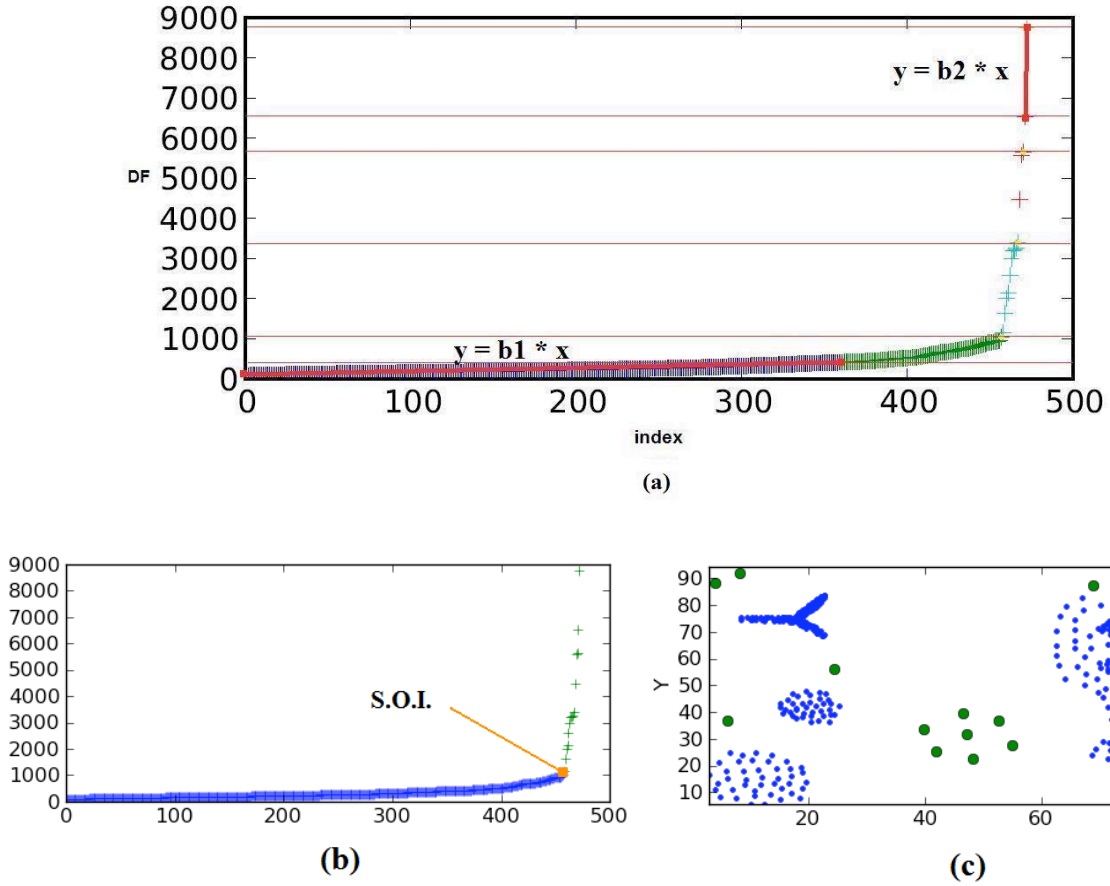


Figura 6. Rilevazione degli outlier tramite l'algoritmo proposto. (a) La procedura approssima i valori *DF* appartenenti al primo strato (valori blu) con una retta di equazione y = b_1*x (linea rossa), e i punti appartenenti all'ultimo strato con una retta di equazione y = b_2*x (linea rossa). La soglia δ, utile per scoprire il primo indice dell'ordinamento da cui considerare gli outlier, viene calcolata come: $\delta = (b_1 + b_2) / 2$. La procedura inizia dal primo indice *i* del secondo strato (valori verdi), calcolando il valore $DF(i) + DF(i+1) / 2$, e si ferma non appena trova una coppia di valori consecutivi ($DF(j), DF(j+1)$), con $j > i$, tali che $DF(j) + DF(j+1) / 2 > \delta$. (b) L'indice *j* verrà considerato come *S.O.I.* (*Start Outlier Index*). (c) Gli outlier vengono rappresentati mediante punti verdi.

1.2 Clustering di dataset con diverse densità di punti

Il clustering ottenuto impiegando l'algoritmo DBSCAN può non essere ottimale quando si lavora su dataset con differenti densità di punti, in quanto in alcuni casi la scelta iniziale dei valori per la coppia di parametri (*MinPts,* ε) può non essere corretta per tutta l'esecuzione dell'algoritmo. In linea di principio, questa limitazione può essere superata eseguendo un settaggio locale dei parametri al posto di uno globale. Sebbene alcuni algoritmi, come ad esempio quelli proposti in Borah et al. (2009) e Fahim et al. (2009), tentino di superare le performances di DBSCAN introducendo alcune ottimizzazioni, hanno il limite di richiedere una buona conoscenza del dataset e il settaggio di un numero di parametri maggiore rispetto a quelli richiesti da DBSCAN.

L'algoritmo ISDBSCAN, implementato in questo software, utilizza l' IS_k*-neighbourhood* [Jin et al., 2006] al posto dell' ε*-neighbourhood* per la valutazione del vicinato per l'espansione dei cluster.

Rispetto alla definizione ε*-neighbourhood* precedentemente introdotta, IS_k*-neighbourhood* è una relazione di vicinato di tipo simmetrico, ovvero punti appartenenti a differenti regioni di densità non possono essere connessi l'un l'altro. Con l'introduzione del concetto di IS_k*-neighbourhood* si può operare una scelta casuale del punto da cui partire con l'espansione del cluster ad esso associato, in quanto le variazioni di densità vengono riconosciute e gestite automaticamente. In questo caso i cluster verranno definiti considerando solo i punti con densità simile e il processo viene reiterato fino a quando tutti i punti con la stessa densità sono stati considerati. La scelta dell'algoritmo è stata incentivata dalla richiesta del settaggio di un solo parametro, k, per il calcolo della IS_k.

2. Implementazione dell'algoritmo DBSCAN

Nel seguente paragrafo verrà descritto il software PyDBSCAN che implementa gli algoritmi discussi nei paragrafi precedenti. L'interfaccia principale mostrata in Fig. 7 contiene una barra dei menu che permette di:

- effettuare il caricamento di files in formato *.tsv* (*tab separated values*), contenenti i dati in colonne separate da un particolare simbolo (spazio, tab, etc.) tramite il comando ***"****File -> Open"*;
- richiamare altri tool (per ora attivo solo il framework *OPTICS*) mediante il comando ***"****Tools"*.

L'intera schermata contiene la maggior parte dei controlli utilizzabili per effettuare il clustering dei dati, il loro pre-processing tramite stratificazione, il settaggio automatico dei parametri e la rimozione del rumore. Tali controlli vengono abilitati solamente dopo il caricamento dei dati.

Il layout dell'interfaccia principale dispone i vari controlli in sei categorie (Fig. 7):

1. *Data Properties* - mostra le proprietà principali del dataset, quali massimo e minimo valore di ogni coordinata dello spazio in cui è definito, numero di oggetti contenuti (*Size*) e diametro dello spazio (*Diameter*);

2. *Plotting File* (Fig. 8): pannello composto da diverse schede, ognuna delle quali utilizzata per visualizzare:
   a. *Dataset*: i punti del dataset (tramite il comando "*Plot"*);
   b. *Layers*: i punti del dataset colorati in maniera differente in base allo strato di appartenenza (gli strati vengono definiti dal processo di stratificazione tramite il comando *"Stratify"*);
   c. *NNk-SumDist*: la curva che rappresenta, per ogni oggetto del dataset, la somma delle distanze dai suoi primi *k* vicini;
   d. *AINFLO*: la curva che restituisce per ogni oggetto il relativo indice di *outlierness*, tramite la relazione derivata da Jin et al. (2006);
   e. *DFk-Partition*: la curva della *Density Function* (*DF*), definita come somma delle due precedenti curve. I valori risultano ordinati in maniera crescente e colorati diversamente a seconda dello strato di appartenenza;

f. *Bipartition*: i punti del dataset, discriminando gli outlier rilevati tramite il comando *"Bipartition"*;
g. *BipartitionCurve*: la curva relativa alla *DF*, restituita dal comando precedente e colorata in modo da distinguere gli outlier dagli oggetti interni;
h. *ISk-Projection*: la proiezione del dataset su una spazio di dimensione maggiore, dove la dimensione in più corrisponde alla somma delle distanze dell'oggetto dai punti appartenenti al suo *ISk* (*k-Influence Space*) [Jin et al., 2006]; per ovvi motivi di visualizzazione, tale plotting viene abilitato solo per dataset con un numero di dimensioni minore di 3;

Figura 7. Interfaccia principale di PyDBSCAN.

3. *Stratification* (Fig. 9) – pannello di controllo del processo di stratificazione. Richiede solamente il settaggio del parametro *k*, utilizzato per il calcolo della NN_k supportato dalla struttura di indicizzazione *kdTree* [Bentley, 1975], il cui codice Python è reso disponibile all'indirizzo http://sites.google.com/site/mikescoderama/Home/kd-tree-knn. L'esecuzione di ogni comando restituisce un feed-back visuale sulla relativa scheda abilitata per il plotting del risultato. Vengono restituiti anche il numero di strati calcolati *"N. Layers"* e il più piccolo indice nell'ordinamento *"S.O.I."* (Start Outlier Index), a partire dal quale la stratificazione classifica gli oggetti come outlier. Tramite il comando *"Save"* si può scegliere se salvare il dataset pre-processato, per poterlo utilizzare come input degli algoritmi di clustering implementati.

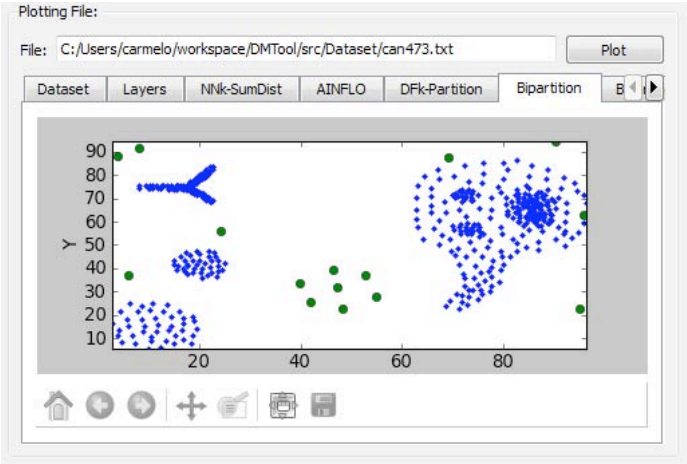


Figura 8. Particolare del pannello *Plotting file* dell'interfaccia utente riportata in Fig. 6.

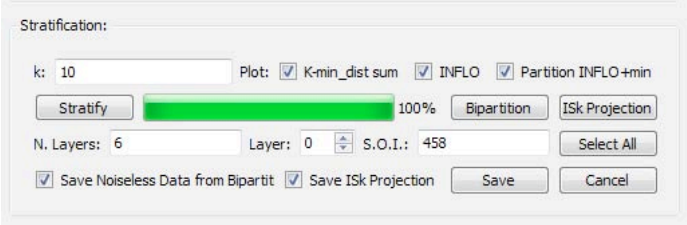


Figura 9. Particolare del pannello *Stratification* dell'interfaccia utente riportata in Fig. 6.

4. *DBSCAN Settings* (Fig. 10): gruppo che contiene i parametri (*MinPts, ε*) da utilizzare nell'algoritmo DBSCAN. Se si impiega la stratificazione come fase di pre-processing, tali valori vengono settati automaticamente dopo l'esecuzione del comando *"Stratify"*. In caso contrario essi possono essere digitati manualmente. Vi sono due diverse modalità di utilizzo dell'algoritmo *"DBSCAN"*: i) è possibile effettuare una sola esecuzione dell'algoritmo tramite il comando *"DBSCAN"* che utilizza come valori la coppia *"Min MinPts","Min Epsilon"*; ii) dopo aver impostato lo step di variazione per *MinPts* ed *ε*, si può effettuare una serie di esecuzioni con tutte le possibili combinazioni tra gli elementi del range [*Min MinPts, Max MinPts*] e [*Min Epsilon, Max Epsilon*].. Per utilizzare quest'ultima tipologia di esecuzione è necessario spuntare la checkbox *"Step by step"*, che abilita il comando *"Next Step"*. I valori dei parametri di un dataset possono essere salvati su file di testo tramite *"Save Params"* e caricati quindi in seguito. Qualora essi fossero ancora presenti nella memoria del programma è possibile caricarli tramite *"Get Params"*, che restituisce gli ultimi valori calcolati dalla stratificazione.

5. *ISDBSCAN Setting* (Fig. 11): permette di utilizzare il metodo ISDBSCAN descritto nel paragrafo 1.2 che dà la possibilità di lavorare anche su dataset che presentano densità spaziali differenti, ovviando ad alcune limitazioni dell'implementazione classica di DBSCAN. E' utile dunque poter configurare il parametro *k* che verrà utilizzato per il calcolo dell'*ISk.* Tale tecnica viene abilitata solo in seguito al calcolo della stratificazione, in quanto essa restituisce i valori necessari per l'esecuzione dell'algoritmo. Viene dunque predisposta anche la scelta della versione del dataset da utilizzare (*"Original Dataset"*, *"Use ISk Projection"*, *"Use Noiseless Dataset"*), per il miglioramento delle performance a livello qualitativo. Sia il risultato di DBSCAN che di

ISDBSCAN può essere salvato su file tramite il comando *"Save"*, all'interno di una cartella identificata dalla coppia [nome dataset, valore parametri]. Ogni cluster sarà identificato da un file, nominato con il suo indice e contenente le coordinate dei punti ad esso associato.

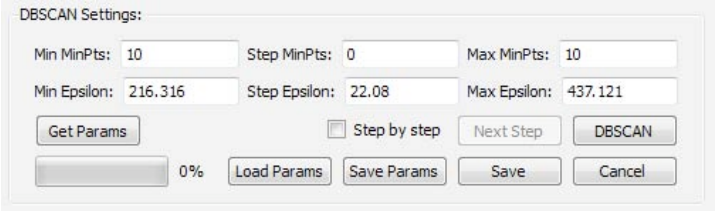


Figura 10. Particolare del pannello *DBSCAN Settings* dell'interfaccia utente riportata in Fig. 6.

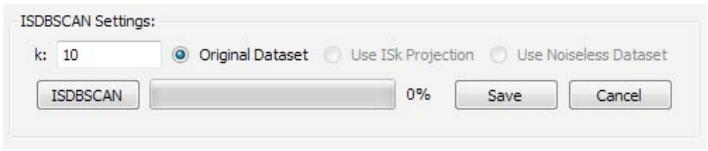


Figura 11. Particolare del pannello *ISDBSCAN Settings* dell'interfaccia utente riportata in Fig. 6.

6. Clustering Output (Fig. 12): pannello di output del clustering effettuato tramite DBSCAN o ISDBSCAN. Il Clustering Output fornisce varie informazioni: a) un plot con i punti del dataset suddivisi in cluster di diverso colore; b) il numero di cluster *"# Clust"*; c) i punti classificati come rumore *"# Noise"*; d) l'indice di validità del clustering, *"DBindex",* ispirato alla formula proposta in Davies e Bouldin (2009); e) i valori dei parametri associati al risultato. E' possibile aprire una finestra di dialogo *"filtering"* (Fig. 13) spuntando su *"Display dialog for filtering"* che predispone alcune funzioni di filtraggio quali la selezione dei cluster principali *"Main Cluster"* di cardinalità maggiore ad una soglia *"% Threshold Cluster Size"*, e la media mobile dei punti visualizzati *"Moving Avg"* utilizzando una finestra di dimensioni *"Moving Avg Window Size"*.

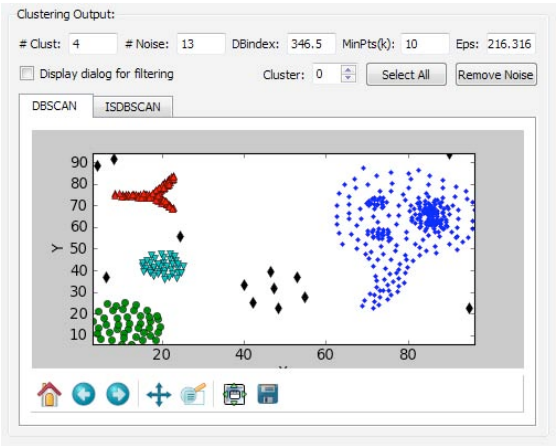


Figura 11. Pannello Clustering Output.

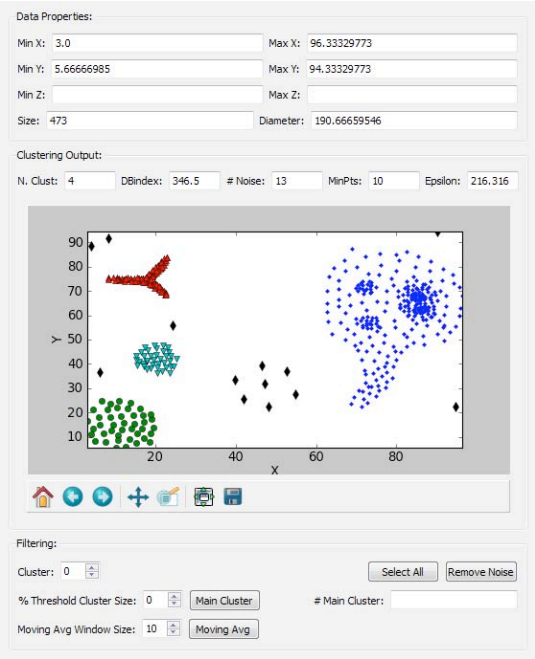


Figura 13. Finestra di dialogo *Filtering*.

3. Risultati ottenuti con il software PyDBSCAN

Nel seguente paragrafo verranno illustrati alcuni risultati ottenuti mediante l'impiego di PyDBSCAN sia su dataset sintetici che su dati reali.

Il primo esempio, riportato in Fig. 14 a,b, mostra l'applicazione su un dataset contente punti appartenenti a clusters di eguale densità all'interno di un insieme rumoroso. Il clustering è stato eseguito mediante l'impiego dell'algoritmo DBSCAN dopo aver settato i parametri ottimali, ricavati dal pre-processamento dei dati basato sulla tecnica della stratificazione. Il risultato del processo è illustrato in Fig. 14b dove i clusters ottenuti sono evidenziati utilizzando differenti colori. Diverso risulta essere il caso mostrato in Fig. 15 a,b, dove il dataset è costituito da un insieme di punti appartenenti a insiemi caratterizzati da diverse densità immersi in uno spazio rumoroso. In questo caso, come descritto nei paragrafi precedenti, l'approccio utilizzato è l'algoritmo ISDBSCAN il cui risultato è riportato in Fig. 15b. In entrambi i test, gli algoritmi implementati si sono rivelati capaci di definire clusters in presenza di rumore.

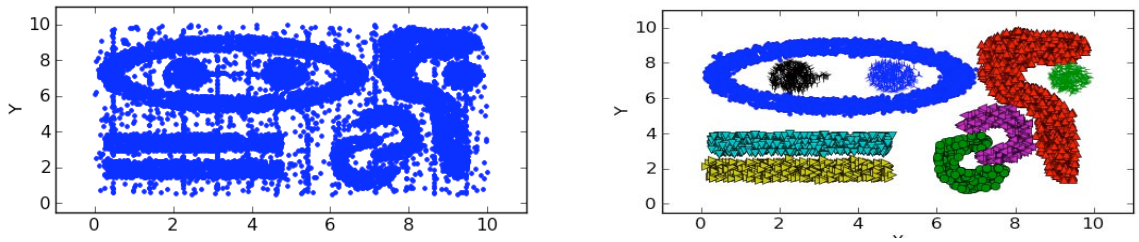


Figura 14. Esempio di clustering eseguito su un dataset sintetico. A sinistra viene riportato lo spazio dei dati non clusterizzati in presenza di rumore.

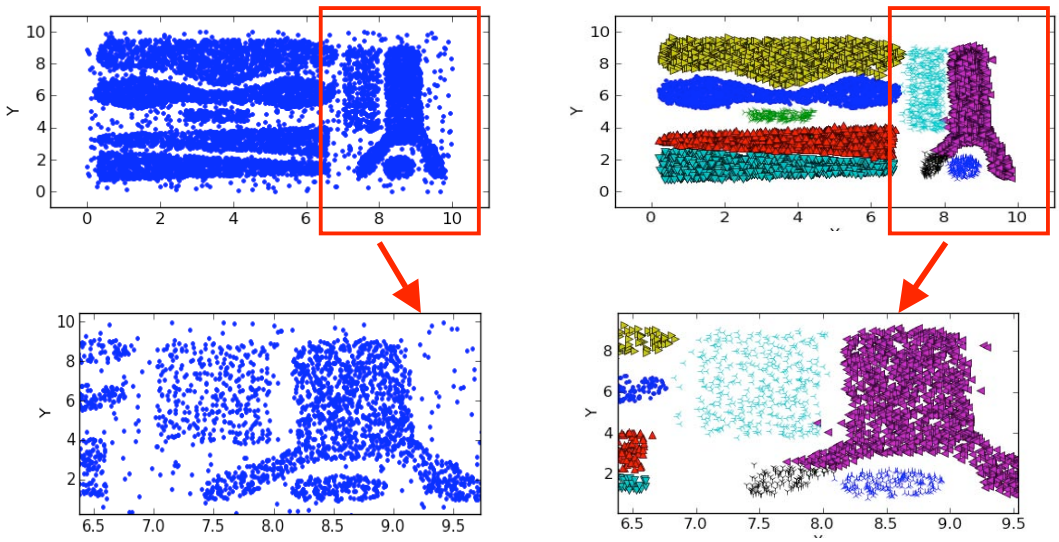


Figura 15. Risultato dell'algoritmo ISDBSCAN. Nei riquadri in rosso viene evidenziata la diversa densità dei punti che costituiscono i clusters.

In ambito di ricerca il software PyDBSCAN è stato impiegato per l'analisi dei segnali infrasonici registrati sul vulcano Etna [Cannata et al., 2011]. In particolare la procedura di clustering è stata eseguita su un dataset (detto spazio delle *features*) costituito da particolari caratteristiche (*features*) descrittive delle forme d'onda associate agli eventi infrasonici come, ad esempio, frequenza di picco, fattore di qualità e ampiezza del transiente infrasonico. Un esempio di spazio dei dati ottenuto e della sua suddivisione in clusters è riportato in Fig. 16 a,b. Come riportato in Cannata et al. (2011), l'analisi dei clusters ottenuti ha permesso l'associazione tra le forme d'onda dei segnali infrasonici, registrati durante il periodo Agosto-Settembre 2007, ed i crateri da cui tali segnali sono stati generati.

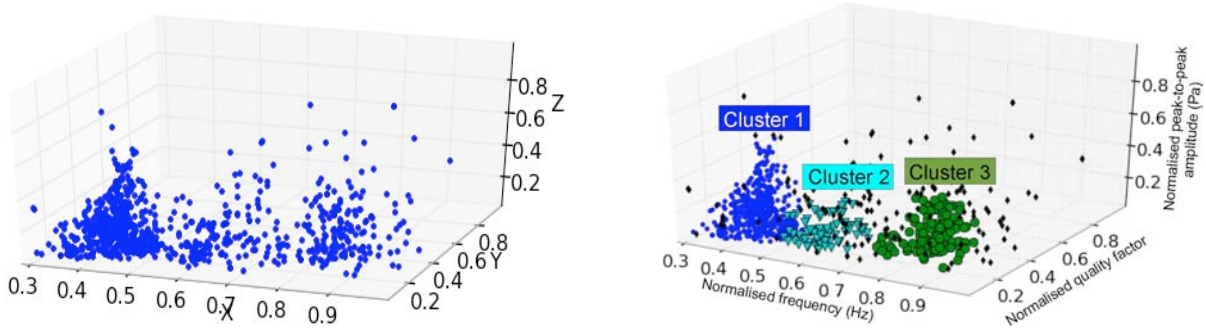


Figura 16. Spazio delle caratteristiche descrittive delle forme d'onda infrasoniche(frequenza di picco, ampiezza e fattore di qualità) registrate sul vulcano Etna nel periodo Agosto – Dicembre 2007. A sinistra viene mostrato lo spazio dei dati prima del processo di clustering. A destra, lo spazio dei dati dopo il clustering. I punti neri indicano elementi etichettati come rumore.

Conclusioni

Nel presente lavoro è stato introdotto il software PyDBSCAN sviluppato presso l'Unità Funzionale Sismologia della sezione di Catania in collaborazione con il Dipartimento di Matematica e Informatica dell'Università degli Studi di Catania. Il programma presentato è basato sulle tecniche di clustering e stratificazione introdotte nel primo paragrafo. La scelta si è focalizzata sulle metodologie basate sul concetto di densità la cui peculiarità sta nell'elevata robustezza al rumore presente nei dati. Questa caratteristica è particolarmente utile quando si lavora con dati, come quelli geofisici, che presentano spesso degli elementi rumorosi che rendono difficili le analisi dei dataset. Il software, sviluppato mediante l'utilizzo del linguaggio di programmazione Python, può essere utilizzato su diverse piattaforme quali Windows e Linux. Dopo la descrizione del software PyDBSCAN, riportata nel secondo paragrafo, sono stati mostrati degli esempi, eseguiti sia su dati sintetici che reali, dove si evidenziano i risultati ottenuti con il software presentato. Sviluppi futuri del prodotto oggetto del report riguarderanno l'integrazione di algoritmi di clustering gerarchico e dinamico.

Bibliografia

Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B., (1990). *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles*. Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ , pp. 322-331.

Bentley, J., (1975). *Multidimensional binary search trees used for associative searching*. Commun. ACM , 18 (9), pp. 509–517.

Borah, B. and Bhattacharyya, D., (2009). *DDSC: A Density Differentiated Spatial Clustering Technique*. Journal of Computers , 3 (2), 72.

Brecheisen, S., Kriegel, H., Kröger, P., Pfeifle, M., (2004). *Visually mining through cluster hierarchies*. In Proc. SIAM Int. Conf. on Data Mining (SDM'04), Lake Buena Vista, FL pp. 400-412.

Breunig, M., Kriegel, H., Raymond, T.Ng, Sander, J., (2000). *LOF: identifying density-based local outliers*. Sigmod Record , 29 (2), pp. 93-104.

Cannata, A., Montalto. P., Aliotta, M., Cassisi, C., Pulvirenti A., Privitera, E., Patanè, D., (2011). *Unsupervised clustering of infrasonic events at Mount Etna using pattern recognition techniques*. Geophys. Jour. Int., in press.

Cantone, D., Ferro, A., Pulvirenti, A., Recupero, D., Shasha, D., (2005). *Antipole tree indexing to support range search and k-nearest neighbor search in metric spaces*. Knowledge and Data Engineering, IEEE Transactions on , 17 (4), pp. 535-550.

Davies, D. and Bouldin, D., (2009). *A cluster separation measure*. (2), pp. 224-227.

Ester, M., Kriegel, H., Sander, J., Xu, X., (1996). *A density-based algorithm for discovering clusters in large spatial databases with noise*. Proc. KDD , 96, pp. 226-231.

Fahim, A., Saake, G., Salem, A., Torkey, F., Ramadan, M., (2009). *Improved DBSCAN for spatial databases with noise and different densities*. Computer, 3, pp. 53-60.

Hartigan, J. and Wong, M., (1979). *A k-means clustering algorithm*. JR Stat. Soc., Ser. C*, 28*, pp. 100-108.

Jin, W., Tung, A., Han, J., Wang, W. (2006). Ranking outliers using symmetric neighborhood relationship. *Advances in Knowledge Discovery and Data Mining*, pp. 577-593.

Matplotlib official site (http://matplotlib.sourceforge.net/)

NumPy official site (http://numpy.scipy.org/)

Python official site (http://www.python.org/).

PyQt (http://www.riverbankcomputing.com/software/pyqt/download)

Coordinamento editoriale e impaginazione
Centro Editoriale Nazionale | INGV

Progetto grafico e redazionale
Daniela Riposati | Laboratorio Grafica e Immagini | INGV

Via di Vigna Murata, 605
00143 Roma
Tel. +39 06518601 Fax +39 065041181

http://www.ingv.it

Istituto Nazionale di Geofisica e Vulcanologia