

Capitolo I

La scoperta del cambiamento nelle immagini telerilevate

1.1 Introduzione

I cambiamenti climatici, il monitoraggio del suolo e la gestione delle emergenze, dovute ad esempio al verificarsi di calamità naturali, sono certamente di interesse comune. A tal proposito il progetto AVIRIS e il più recente programma Copernicus, che vede coinvolti vari enti europei, forniscono una grande mole di dati che possono essere analizzati, mediante l'utilizzo delle tecnologie informatiche, al fine di studiare le tematiche in oggetto.

Il programma Copernicus prevede sei missioni:

- Sentinel-1 fornisce dati meteorologici e immagini radar per il monitoraggio del suolo e dei mari. Il 3 aprile 2014 il primo satellite Sentinel-1A è stato lanciato mentre il suo satellite gemello ha preso parte alla missione il 25 aprile 2016.
- Sentinel-2 fa utilizzo di due satelliti identici, denominati Sentinel-2A e Sentinel-2B, lanciati rispettivamente il 23 giugno 2015 e il 7 marzo 2017. I due satelliti sono sfasati di 180° e viaggiano ad un'altitudine media di 786 km. Entrambi

trasportano uno strumento per la rilevazione di radianza multispettrale. Questi strumenti sono in grado di rilevare 13 bande a diversa risoluzione spaziale (dai 10 ai 60 m). La copertura geografica include il mar mediterraneo, le isole dell’Unione Europea e tutte le isole con lunghezza costiera non superiore ai 20 km. I dati pubblicati sono differenti fra loro: quelli di livello 1b contengono registrazioni riguardo la parte superiore dell’atmosfera, i dati di livello 1c contengono anch’essi registrazione della parte superiore dell’atmosfera ma sono radiometricamente e geometricamente corretti; i dati di livello 2a forniscono registrazione della parte inferiore dell’atmosfera e sono corretti radiometricamente e geometricamente.

- Sentinel-3 fornisce dati come la topografia, la temperatura e il colore della superficie del mare e della terra. Il primo satellite è stato lanciato il 16 febbraio 2016, il secondo è stato lanciato il 25 aprile 2018.
- Sentinel-4 è atta al monitoraggio dell’atmosfera, in particolare dell’ozonosfera. La copertura geografica comprende l’Unione Europea, il Nord Africa e parte dell’oceano Atlantico. Tra dati che saranno forniti quelli di livello 1b saranno radiometricamente e spettralmente calibrati. Ne deriveranno dati contenti informazioni sulla copertura nuvolosa e sui gas presenti nell’atmosfera. Tali rilevazioni saranno possibili grazie a due spettrometri multispettrali che opereranno nelle bande dell’ultravioletto, visibile e prossima all’infrarosso, con risoluzione spaziale di 8 km.
- Sentinel-5 monitorerà la qualità dell’aria e le interazioni che si verificano tra il clima e la composizione chimica dell’atmosfera. Cinque spettrometri saranno impiegati con lo scopo di effettuare rilevazioni riguardanti la parte superiore dell’atmosfera. Opereranno nello spettro UVNS (ultravioletto, visibile, prossimo all’infrarosso, infrarosso ad onde corte) con una risoluzione spaziale di 50 km per la prima parte dell’ultravioletto (270-300nm) e 7.5 km per le altre bande. Un sottosistema di calibrazione sarà in grado di calibrare lo strumento in base alla

posizione e allo stato del Sole. I dati forniti di livello 1b saranno geolocalizzati e corretti radiometricamente. Il lancio è previsto per il 2021.

- Sentinel-5P ha lo scopo di monitorare la qualità dell'aria, l' ozonosfera e studiare la relazione tra il cambiamento climatico e il metano presente nell'atmosfera. Lo strumento di monitoraggio della troposfera, che orbiterà ad una altitudine di 824 km, effettuerà registrazioni nello spettro UVNS (ultravioletto, visibile, prossimo all'infrarosso, infrarosso ad onde corte) con una risoluzione spaziale di 7 x 3,5 km. I dati saranno pubblicati, geolocalizzati e corretti radiometricamente, entro 12 ore dal rilevamento. Il lancio è avvenuto il 13 ottobre 2017.

1.2 Telerilevamento

Il telerilevamento è l'osservazione, fatta senza contatto diretto, delle proprietà di un oggetto attraverso i segnali elettromagnetici che esso produce in conseguenza della sua forma e del suo stato.

I sensori spettrali, nello specifico, sono sensori ottici in grado di produrre immagini contenenti, per ogni pixel, misurazioni della radianza spettrale del suolo in canali spettrali (o bande spettrali) contigui, aventi ognuno differente lunghezza d'onda. Lo sviluppo di questi sensori deriva dalla convergenza di due ambiti: la spettroscopia e il remote imaging.

La spettroscopia è lo studio della luce emessa o riflessa dai materiali e la sua variazione in energia rispetto alla lunghezza d'onda, mentre i remote imager sono strumenti progettati al fine di misurare la luce riflessa da aree adiacenti della superficie terrestre, a cui ci si riferisce con il termine radianza.

Le immagini prodotte dai remote imager hanno una risoluzione spaziale che dipende dall'altitudine in cui le apparecchiature si trovano al momento dello

scatto. Possiamo interpretare la risoluzione spaziale come la dimensione degli appezzamenti rilevati in ogni pixel. Oggetto di studio delle immagini telerilevate è la riflettanza spettrale, definita come il rapporto tra l'energia riflessa e l'energia incidente relative ad una lunghezza d'onda fissata. Considerando che i remote imager rilevano la radianza spettrale, nel caso di immagini raffiguranti la superficie terrestre, vanno tenuti presente alcuni fattori quali: la posizione del sole, gli effetti atmosferici e gli errori commessi dal sensore stesso al fine di poter confrontare le registrazioni effettuate.

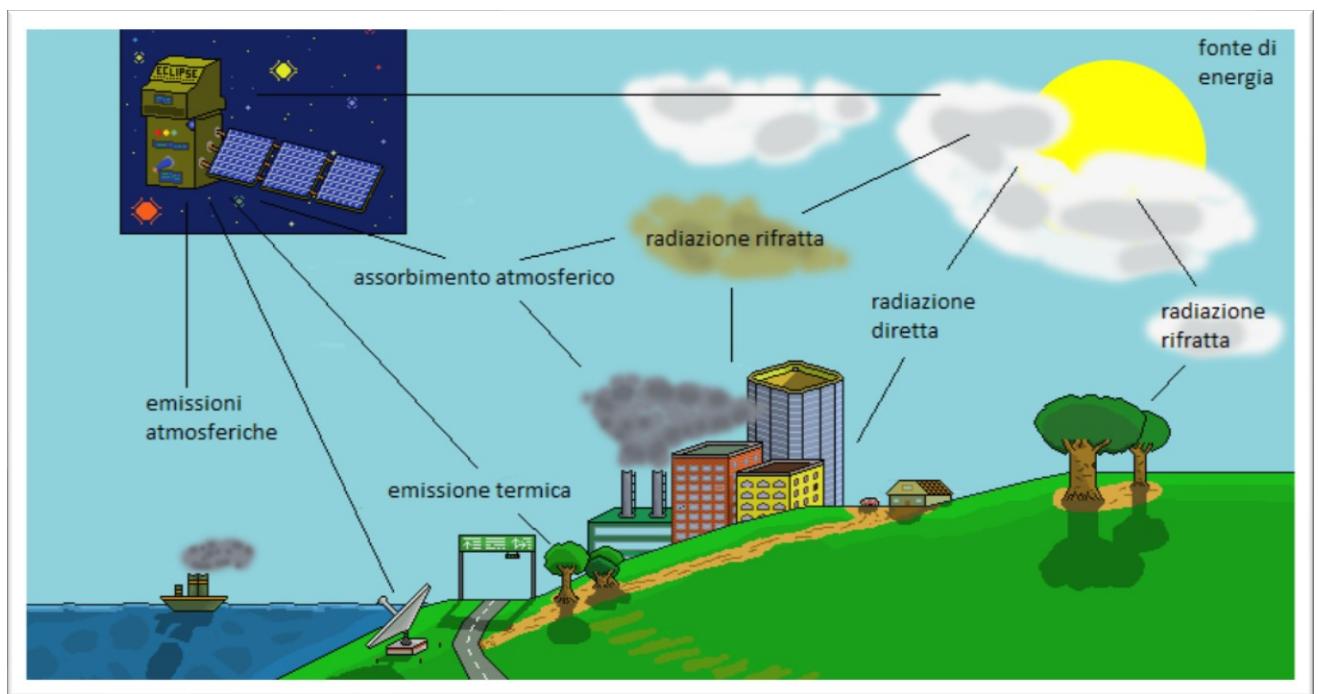


Figura 1.1 Il sole emette delle radiazioni che possono raggiungere la superficie terrestre in modo perpendicolare oppure a seguito di una rifrazione dovuta alla presenza di composti chimici presenti nell'atmosfera (nuvole, inquinamento, ...). La materia presente nella scena - vegetazione, edifici ma anche composti chimici che fluttuano nell'atmosfera - riflettono queste radiazioni e i remote sensor sono in grado di rilevare l'intensità delle radiazioni riflesse. Questi "dati grezzi" sono inviati sulla Terra e subiscono una elaborazione prima di essere pubblicati.

Una prima fase di elaborazione consiste nella trasformazione della riflettanza spettrale in radianza spettrale. (Figura 1.1) Questo può avvenire tramite strategie che sfruttano conoscenza, come le condizioni atmosferiche al momento della rilevazione oppure tramite strategie che sfruttano la sola informazione contenuta nell'immagine rilevata. Nell'ambito del telerilevamento si distinguono le

immagini multispettrali, dove le misurazioni di riflettanza si riferiscono ad un numero esiguo di bande, generalmente non più di 10, dalle immagini iperspettrali, dove i canali spettrali sono in numero maggiore, indicativamente più di 100 (Figura 1.2). [6]

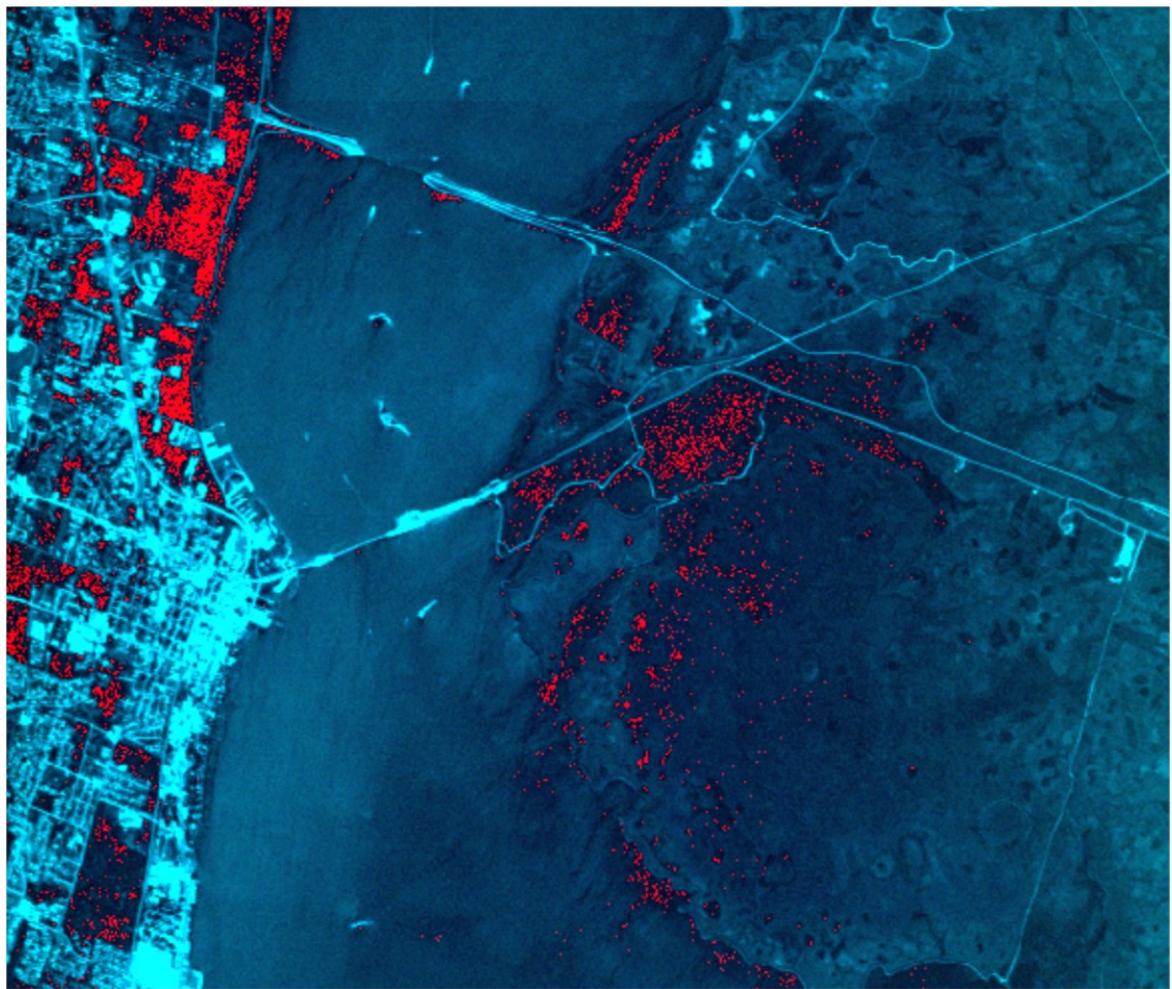


Figura 1.2 Immagine iperspettrale telerilevata dallo strumento AVIRIS di proprietà della NASA, raffigurante lo spazioporto John F. Kennedy Space Center, Florida, USA.

1.3 Rilevare il cambiamento

La rilevazione del cambiamento è il processo che ci permette di identificare differenze nello stato di un oggetto, o fenomeno, osservato in differenti istanti di tempo.

Al fine di rilevare il cambiamento viene sfruttata la corrispondenza tra i cambiamenti degli oggetti presenti nella scena telerilevata e la variazione di radianza emessa da ognuno di essi; auspicando che la variazione di radianza dovuta al cambiamento sia significativamente maggiore della variazione di radianza dovuta a fattori esterni.

È necessario, infatti:

- tenere conto delle condizioni atmosferiche, dell'angolazione del sole e dell'umidità del suolo, che influiscono sulla radianza dei materiali. È bene quindi utilizzare immagini registrate impiegando lo stesso sensore, con stessa risoluzione spaziale, nello stesso periodo dell'anno; [1]
- che le immagini acquisite in diversi istanti di tempo immortalino la stessa scena, si richiede quindi che queste siano perfettamente sovrapponibili;
- che le registrazioni siano temporalmente coerenti con il fenomeno oggetto di studio. [7]

Nel corso degli anni sono emersi vari approcci al problema della change detection, questi possono essere classificati in:

- Pixel-based: le caratteristiche spettrali di ogni pixel sono sfruttate senza considerarne il contesto spaziale;
- Object-based: in cui il cambiamento viene rilevato confrontando la segmentazione delle immagini in oggetti raffigurati dalle stesse. [4]

Le tecniche di change detection, generalmente, producono informazione latente (ad esempio un'immagine derivata da un'operazione effettuata su altre immagini) tramite cui è possibile stabilire l'eventuale presenza di cambiamenti. Per individuare tali cambiamenti è necessario stabilire un valore soglia; nel caso in cui il valore codificato nell'informazione latente sia superiore a questo, evidenzieremo la presenza di un cambiamento.

Nella tecnica di “image differencing” date due immagini raffiguranti la stessa scena in due differenti istanti di tempo, si costruisce una terza immagine risultante dall’operazione di sottrazione tra i pixel delle due immagini iniziali.

L’immagine ottenuta, per ogni banda di ogni pixel, definisce una distribuzione empirica. Di conseguenza, dato il valore di una banda di un dato pixel, rileveremo il cambiamento se questo valore si stabilisce sulle code di questa distribuzione. In altre parole possiamo usare, come soglia, valori di deviazione standard.

Esempio

Consideriamo i seguenti campioni casuali, generati da una distribuzione multinormale con media:

$$[1010, 820, 698, 1200, 1020, 2002, 1890]^T$$

E matrice di varianze e covarianze:

$$[7, 0, 0, 0, 0, 0, 0]$$

$$[0, 3, 0, 0, 0, 0, 0]$$

$$[0, 0, 4, 0, 0, 0, 0]$$

$$[0, 0, 0, 9, 0, 0, 0]$$

$$[0, 0, 0, 0, 6, 0, 0]$$

$$[0, 0, 0, 0, 0, 11, 0]$$

$$[0, 0, 0, 0, 0, 0, 12]$$

Il data set al tempo t0 è:

$$[1011.75773256, \quad 820.48125838, \quad 700.74781676, \quad 1205.83791214, \\ 1019.13221263, 2003.28479485, 1890.71385111]$$

$$[1007.33428178, \quad 818.46742325, \quad 699.11039103, \quad 1201.42942237, \\ 1018.86906257, 2003.41101684, 1890.21250398]$$

[1005.03756169, 818.65473743, 697.70711611, 1206.26316848,
1023.36098743, 2001.56644269, 1891.09578546]

Il data set al tempo t1 è:

[1010.40840609, 822.89988018, 696.0925452, 1204.8520043, 1018.74606512,
2005.52391073, 1893.57109619]

[1010.46823389, 817.687703, 699.96888051, 1201.25528083, 1023.68091644,
2001.62270907, 1885.97705234]

[1004.03756169, 818.15473743, 697.00711611, 1206.06316848,
1023.06098743, 2001.00644269, 1891.00578546]

Il data set differenza risulta essere:

[1.34932647, -2.4186218, 4.65527155, 0.98590784, 0.38614751, -2.23911588, -
2.85724508]

[-3.13395211, 0.77972025, -0.85848948, 0.17414154, -4.81185387, 1.78830777,
4.23545164]

[-1.00000000, -0.50000000, -0.70000000, -0.20000000, -0.30000000, -
0.56000000, -0.09000000]

I valori di deviazione standard per ogni componente indipendente sono:

2.65, 1.73, 2, 3, 2.45, 3.32, 3.46

Quindi, il primo pixel evidenzia un cambiamento in quanto il valore assoluto della seconda e terza componente superano il valore di deviazione standard. In modo simile il secondo pixel evidenzia un cambiamento in quanto il valore assoluto della prima, quinta e settima componente superano il valore di deviazione standard.

Il terzo pixel invece non evidenzia alcun cambiamento.

Nella tecnica di “image ratioing”, due immagini registrate in istanti di tempo differenti sono utilizzate per costruire una nuova immagine dove, per ogni pixel, il valore di una banda è calcolato come il rapporto del valore della stessa banda nelle due immagini di partenza.

L’immagine risultante viene quindi utilizzata per stabilire la presenza di cambiamenti: bande aventi valori significativamente più grandi o più piccoli di 1 evidenzieranno la presenza di un cambiamento.

Notiamo che le distribuzioni di probabilità delle bande nelle immagine risultante dall’operazione di ratioing non seguono una distribuzione normale, non è possibile, quindi, derivare una soglia tramite la deviazione standard perché, così facendo, otterremmo un criterio asimmetrico che privilegia valori maggiori di 1 rispetto a valori minori di 1, o viceversa. [1]

I valori di radianza spettrale possono essere analizzati non solo considerando singole bande ma anche utilizzando più bande contemporaneamente. Ad esempio, nell’ambito degli studi sulla vegetazione sono state definite trasformazioni di dati -che prendono il nome di indici- relazionabili con la presenza di organismi vegetali a cui ci si riferisce tipicamente con l’espressione biomassa verde. Il più comune di questi è l’indice di vegetazione normalizzato (NVI, normalized vegetation index) che si ottiene mediante la seguente espressione:

$$NVI = \frac{\text{banda prossima agli infrarossi} - \text{banda rossa}}{\text{banda prossima agli infrarossi} + \text{banda rossa}}$$

Ne deduciamo la possibilità di trasformare un’intera immagine sostituendo ad ogni pixel il corrispondente valore di vegetazione normalizzato. [1] In seguito, per stabilire la presenza di un cambiamento nel quantitativo di vegetazione, è possibile applicare uno dei metodi precedenti partendo dalle immagini trasformate. Un altro indice molto utilizzato è l’indice di terreno spoglio (BSI, bare soil index) che può essere calcolato con la formula: [2]

$$= \frac{((banda\ infrarossi\ ad\ onde\ corte + banda\ rossa) - (banda\ prossima\ agli\ infrarossi + banda\ blu))}{((banda\ infrarossi\ ad\ onde\ corte + banda\ rossa) + (banda\ prossima\ agli\ infrarossi + banda\ blu))}$$

Nei casi in cui le registrazioni effettuate presentano sostanziali differenze nella situazione atmosferica, ad esempio la presenza di nubi, nella scena può essere utile applicare la “analisi delle componenti principali”. Con questa tecnica, due registrazioni vengono sovrapposte, ottenendo così un’immagine contenente in ogni pixel una quantità doppia di valori di riflettanza. Da quest’ultima immagine generiamo un numero k di immagini, ognuna di questa è ottenuta proiettando le bande spettrali contenute in ogni pixel sull’i-esima componente principale (per $i=1,\dots,k$). Le immagini relative alle componenti principali associate agli autovalori più grandi, della matrice di varianze e covarianze campionarie dell’immagine frutto della sovrapposizione, conterranno l’informazione riguardante il cambiamento dovuto alle differenze atmosferiche. Queste ultime portano a registrazioni di radianza molto differenti nel tempo, mentre le componenti principali associate agli autovalori più piccoli conterranno l’informazione necessaria a rilevare il cambiamento sul suolo della scena, utilizzando una delle tecniche precedenti.[1]

In alcuni casi, l’analisi che si vuole condurre, è sia di tipo qualitativo che quantitativo. Per farlo è possibile utilizzare la tecnica che ci fornisce un vettore di cambiamento nell’iperspazio, come risultato dell’operazione di differenza tra una coppia di pixel, la cui dimensionalità è definita dal numero di bande registrate. Possiamo interpretare la lunghezza di questo vettore come una misura quantitativa del cambiamento. È possibile applicare questa tecnica, nota come “change vector analysis”, a seguito di una fase di pre-processazione delle immagini, possiamo trasformare le immagini iniziali usando la tecnica degli indici. Applicando a questa la change vector analysis, l’angolo del vettore di cambiamento ci permette un’interpretazione di tipo “da a” (ad esempio “da terreno spoglio a vegetazione”).

[1]

Nel caso in cui le immagini abbiano risoluzione spaziale bassa (inferiore ai 20 m^2) non è più possibile trascurare il fatto che i valori di riflettanza associati ad un pixel siano frutto dei valori di radianza acquisita di più materiali.

Lo “spectral unmixing” è il processo di scomposizione dei pixel in una sequenza di fattori di abbondanza relativi a firme spettrali (endmember) di materiali presenti nella scena in forma pura. Questa tecnica non solo ci permette di rilevare il cambiamento quantitativo dei materiali presenti nella scena, ma anche, l’aggiunta o detrazione di un materiale. La strategia consiste di quattro passaggi:

1. Rilevare la dimensionalità dei dati, ovvero la quantità di materiali distinti presenti nella scena;
2. Estrarre per ognuno di questi materiali il suo endmember;
3. Calcolare le mappe di abbondanza di ogni pixel per ogni istante di tempo registrato;
4. Rilevare il cambiamento nel tempo sfruttando le mappe di abbondanza. [3]

Il primo passaggio può essere effettuato utilizzando algoritmi noti in letteratura per l’identificazione di segnali. Partendo quindi dallo spazio delle bande spettrali otterremo un sottospazio teoricamente privo di rumore. [4] La dimensionalità di questo equivarrà alla quantità di materiali distinti presenti nella scena. Per l’estrazione degli endmember dalla scena è possibile utilizzare algoritmi noti che possono assumere la presenza dei materiali in forma pura (ad esempio N-FINDR) o meno (ad esempio SISAL). È possibile ora calcolare i fattori di abbondanza di ogni pixel, rispetto agli endmember estratti, utilizzando il modello lineare. I valori ottenuti sono quindi utilizzati per ricavare, per ogni registrazione, un numero di mappe di abbondanza pari al numero degli endmember rilevati. La k-esima mappa di abbondanza avrà come valore assegnato al pixel in posizione (i,j) il fattore di abbondanza calcolato per il k-esimo endmember. Quindi, sarà possibile applicare a coppie di mappe di abbondanza riguardanti la stessa scena, in istanti di tempo

differenti, la tecnica di image differencing per stabilire la presenza di cambiamento. [1]

Le tecniche finora discusse afferiscono al paradigma pixel-based. Alcuni fattori hanno portato allo sviluppo del paradigma object-based. Questi principalmente sono:

- Spaziali: l'approccio pixel-based ci costringe ad effettuare analisi utilizzando una singola risoluzione spaziale. E' impossibile definire una risoluzione spaziale che permetta di effettuare analisi accurate per tutti i tipi di oggetti presenti nella scena. L'approccio object-based risolve questo problema effettuando segmentazioni multiscala. Gli algoritmi di segmentazione infatti utilizzano uno parametro di scala che stabilisce la grandezza media, minima e massima dei segmenti risultanti. Ad esempio, impostando questo parametro ad un valore basso potremo delineare oggetti di piccole dimensioni, ad esempio case; al contrario, valori alti possono essere utilizzati per segmentare appezzamenti agricoli.
- Alta variabilità spettrale: può accadere che un'immagine multitemporale (immagini raffiguranti la stessa scena ma registrate in differenti istanti di tempo) presenti un'alta variabilità spettrale, per una o più bande spettrali, in alcuni pixel. Questo può essere dovuto ad esempio alla presenza di un'abbondante copertura nuvolosa durante una registrazione. Applicare il paradigma pixel-based in questi casi porta alla rilevazione di tanti piccoli cambiamenti spuri. Nel gergo ci si riferisce a questo con l'espressione “salt and pepper effect”.

Durante la fase di segmentazione le immagini sono partizionate in aree adiacenti dette segmenti, dove, pixel adiacenti sono accorpati ad uno stesso segmento se sufficientemente omogenei. [5] I segmenti ottenuti sono utilizzati in rappresentanza di oggetti del mondo reale e questa associazione avviene in base agli attributi spettrali (ad esempio il valore medio assunto da una banda spettrale) e spaziali (ad esempio le proprietà geometriche dell'oggetto: forma, area). La criticità dell'approccio object-based risiede nella ricerca di oggetti corrispondenti

nelle immagini multitemporali in quanto gli oggetti che subiscono un cambiamento modificano i loro attributi.

Nell'ambito dell'object-based possiamo effettuare un'ulteriore classificazione, si parla infatti di image-object e di class-object. Nell'approccio image-object i cambiamenti sono rilevati sfruttando direttamente le informazioni spettrali e spaziali di un oggetto raffigurato nell'immagine multitemporale oggetto di analisi.

Nell'approccio class-object il cambiamento è invece rilevato confrontando gli oggetti presenti nella scena, a seguito della classificazione degli stessi. Si riesce così ad ottenere un'informazione di tipo “da a”.[5]

Capitolo II

Analisi dei gruppi

In questo capitolo si descrive il problema del clustering ed in particolare l'algoritmo K-means. Successivamente si introduce il framework per il calcolo distribuito Apache Hadoop, il modello computazionale MapReduce e la sua implementazione Hadoop MapReduce. Infine si descrive una possibile implementazione di K-means che sfrutta MapReduce.

2.1 Introduzione

Considerato un data set di n osservazioni, o punti, di un vettore casuale p -dimensionale X , avente funzione di densità di probabilità congiunta $P(X)$, si vogliono inferire le proprietà di questa funzione di densità.

L'analisi dei gruppi (o cluster), tipicamente, consiste nel trovare più regioni convesse, nello spazio in cui giace X , ognuna delle quali contente una moda di $P(X)$. Questo al fine di rappresentare la densità $P(X)$ come un insieme di densità di probabilità note (tipicamente Gaussiane), ognuna associata ad gruppo distinto di osservazioni.

Nel contesto dell'apprendimento non supervisionato, e quindi anche nel caso specifico del clustering, non c'è modo di stabilire, analiticamente, con certezza se

il risultato ottenuto è di buona fattura. Tuttavia alcuni indici di valutazione sono utilizzati, ad esempio: [8]

1. Indice di Silhouette: ci permette di stabilire se si è raggiunto l'obbiettivo di massimizzare la dissimilarità fra punti appartenenti a cluster diversi e minimizzare la dissimilarità fra punti appartenenti allo stesso cluster. Questo indice, nello specifico, usa delle considerazioni geometriche quali separazione e compattezza per stabilire la qualità del risultato ottenuto dall'operazione di clustering. L'indice di Silhouette assume valori in [-1,1], dove -1 indica una bassa compattezza e valori vicini ad 1 indicano un'alta separazione ed un'alta compattezza.

$$Silhouette = \frac{1}{n} \sum_{i=1}^n s_i \text{ con } s_i = \frac{b_i - w_i}{\max\{b_i, w_i\}}$$

Dove n è la cardinalità del data set, b_i è la più piccola delle medie delle distanze tra il punto x_i e i punti in uno stesso cluster diverso da quello di appartenenza di x_i , w_i è la media delle distanze fra x_i e i punti nello stesso cluster di x_i .

2. Indice di Rand: facendo utilizzo di una ground truth (dati correttamente annotati) ci permette di effettuare un confronto di consistenza con il risultato dell'operazione di clustering. Il confronto avviene considerando coppie di dati e verificando i loro cluster di appartenenza nei due casi. Questo indice assume valori in [0,1], dove 0 indica totale inconsistenza e 1 indica totale consistenza.

$$Rand\ Index = \frac{a + d}{\binom{n}{2}}$$

Dove n è la cardinalità del data set, a è il numero di coppie presenti nella stessa classe della ground truth e nello stesso cluster, d è il numero di coppie i cui elementi sono di classe differente nella ground truth e appartengono a cluster diversi.

Questi, in ogni caso, non sostituiscono la verifica diretta del risultato da parte dell'analista, il quale sfrutta la conoscenza di dominio.

L'analisi dei gruppi consiste nel raggruppamento o nella segmentazione di una collezione di oggetti in più sottoinsiemi (o cluster), in modo tale che, gli oggetti presenti in uno stesso cluster sono molto più relati fra loro di quanto lo siano oggetti appartenenti a cluster diversi.

Talvolta l'obiettivo di tale analisi è organizzare i cluster risultanti in una gerarchia. Si arriva a questo raggruppando ricorsivamente i cluster stessi, in modo tale che ad ogni livello della gerarchia, i cluster nello stesso gruppo sono più simili tra loro rispetto a cluster appartenenti a gruppi differenti.

In altri casi l'obiettivo è approssimare un insieme infinito sfruttando un insieme finito i cui elementi sono detti rappresentanti. Si parla di vector quantization.

I metodi di clustering raggruppano gli oggetti presenti nel data set di partenza, sfruttando la definizione di similarità o dissimilarità fornita.

A volte i dati sono espressi direttamente in termini di affinità (similarità o dissimilarità) fra coppie di oggetti. Questi possono essere rappresentati da una matrice D di dimensione $n \times n$, dove ogni elemento d_{ij} registra l'affinità tra l' i -esimo e il j -esimo oggetto. La maggior parte degli algoritmi prevedono che questa matrice abbia gli elementi diagonali pari a 0 e i restanti positivi. Notiamo che, le affinità soggettive, direttamente espresse da esperti di dominio, non sono distanze; infatti la diseguaglianza triangolare $d_{ij} \leq d_{ik} + d_{jk}$ non risulta essere vera in generale.

Possiamo categorizzare le variabili coinvolte nel processo di clustering come: quantitative (che assumono valori nell'insieme dei numeri reali o in sottoinsiemi di esso), ordinali (che assumono valori interi contigui) e categoriche (che assumono valori in un insieme non ordinato).

Gli algoritmi di clustering possono essere di tre tipi:

- Non parametrici: non vi è alcuna assunzione distributiva alla base del loro funzionamento;
- Parametrici: un'assunzione distributiva (o un modello probabilistico) è alla base del loro funzionamento;
- Mode seekers: sono non parametrici ma cercano di stimare le mode presenti nella funzione di densità di probabilità (questa è tipicamente multimodale). Le mode sono rappresentanti di cluster e le osservazioni sono associate ad un cluster al posto che ad un altro in base alla moda più “vicina”. [8]

2.2 K-means

L'algoritmo K-means è un algoritmo di clustering non parametrico, applicabile quando le osservazioni sono n-pie di variabili quantitative continue, con l'obiettivo di approssimarle utilizzando un insieme finito di rappresentanti. La dissimilarità tra le osservazioni (o oggetti) del data set sono espresse mediante la distanza euclidea quadratica, o distanza L_2^2 . Le osservazioni risultano essere punti nello spazio metrico (R^d, L_2^2) .

Sia $S \subset R^d$ il data set di partenza.

La distorsione di $x \in S$ è la distanza tra x ed il rappresentante di cluster ad x più vicino.

L'obiettivo è minimizzare la distorsione tipica (o media) dei punti nel data set S . Formalmente:

K-MEANS CLUSTERING

Input: Insieme finito $S \subset R^d$, intero k .

Output: $T \in R^d$ con $|T| = k$.

Obiettivo: Minimizzare $cost(T) = \sum_{x \in S} \min_{z \in T} \|x - z\|^2$.

I rappresentanti in T inducono una partizione di Voronoi di R^d ; ovvero una partizione fatta di k insiemi convessi, ognuno corrispondente ad un rappresentante z , contenente la regione di spazio avente z come rappresentante più vicino.

Questa partizione induce un clustering ottimale (locale) $S = \bigcup_{z \in T} C_z$, dove:

$$C_z = \{x \in S \mid \text{il rappresentante più vicino ad } x \text{ è } z\}$$

La funzione di costo può essere equivalentemente riscritta come:

$$cost(T) = \sum_{z \in T} \sum_{x \in C_z} \|x - z\|^2.$$

Considerata una qualsiasi partizione di Voronoi fatta dei cluster C_1, \dots, C_k , con rappresentanti z_1, \dots, z_k esprimiamo la funzione di costo come segue:

$$cost(C_1, \dots, C_k; z_1, \dots, z_k) = \sum_{j=1}^k \sum_{x \in C_j} \|x - z_j\|^2.$$

Considerato un singolo cluster C con rappresentante z , il costo ad esso associato è:

$$cost(C; z) = \sum_{x \in C} \|x - z\|^2.$$

Questo è minimizzato quando $z = \text{mean}(C)$.

Lemma 1. Per un qualsiasi insieme $C \subset R^d$ ed un qualsiasi $z \in R^d$:

$$cost(C; z) = cost(C; \text{mean}(C)) + |C| * \|z - \text{mean}(C)\|^2$$

Questo teorema deriva dalla decomposizione distorsione-varianza (bias-variance) dei vettori casuali.

Lemma 2. Sia $X \in R^d$ un qualsiasi vettore casuale. Per ogni $z \in R^d$:

$$E||X - z||^2 = E||X - EX||^2 + ||z - EX||^2$$

Dimostrazione. Infatti se X è un campione casuale in C :

$$E||X - z||^2 = \sum_{x \in C} \frac{1}{|C|} |x - z|^2 = \frac{1}{|C|} cost(C; z)$$

E

$$E||X - EX||^2 = \frac{1}{|C|} cost(C; mean(C))$$

L'algoritmo che risolve il problema K-means, anch'esso detto K-means, è il seguente:

Inizializza i centri $z_1, \dots, z_k \in R^d$ e i cluster C_1, \dots, C_k

do

for each $1 \leq j \leq k$: $C_j \leftarrow \{x \in S \mid \text{il centroide più vicino ad } x \text{ è } z_j\}$

for each $1 \leq j \leq k$: $z_j \leftarrow mean(C_j)$

while non ci sono cambiamenti in $cost(C_1, \dots, C_k, z_1, \dots, z_k)$

La complessità in tempo dell'algoritmo è $O(k|S|)$ per iterazione.

Teorema. Durante l'esecuzione di K-means, il costo è una funzione monotona decrescente.

Dimostrazione. Siano $z_1^t, \dots, z_k^t, C_1^t, \dots, C_k^t$ i k centroidi e i k cluster all'inizio della t -esima iterazione di K-means. Siccome il primo passo del ciclo do...while assegna ogni punto al centro più vicino:

$$cost(C_1^{t+1}, \dots, C_k^{t+1}, z_1^t, \dots, z_k^t) \leq cost(C_1^t, \dots, C_k^t, z_1^t, \dots, z_k^t)$$

Tenendo conto che, il secondo passo del ciclo do...while centra ogni cluster rispetto alla sua nuova media (o centroide), per il Lemma 1 precedente:

$$cost(C_1^{t+1}, \dots, C_k^{t+1}, z_1^{t+1}, \dots, z_k^{t+1}) \leq cost(C_1^t, \dots, C_k^t, z_1^t, \dots, z_k^t). [10]$$

2.3 Apache Hadoop

La libreria Apache Hadoop è un framework che permette di processare data set di grandi dimensioni, operando mediante computazione distribuita sfruttando cluster di computer. Hadoop permette di fornire la così detta high-availability utilizzando hardware comune, essendo progettato per rilevare e gestire i fallimenti a livello software.

Hadoop utilizza una architettura a tre livelli, che per quanto riguarda il mio lavoro, è riducibile a quanto segue:

- Livello Application: MapReduce
- Livello Compute: Yarn (Yet Another Resource Negotiator)
- Livello Storage: Hdfs (Hadoop Distributed Filesystem)

[11]

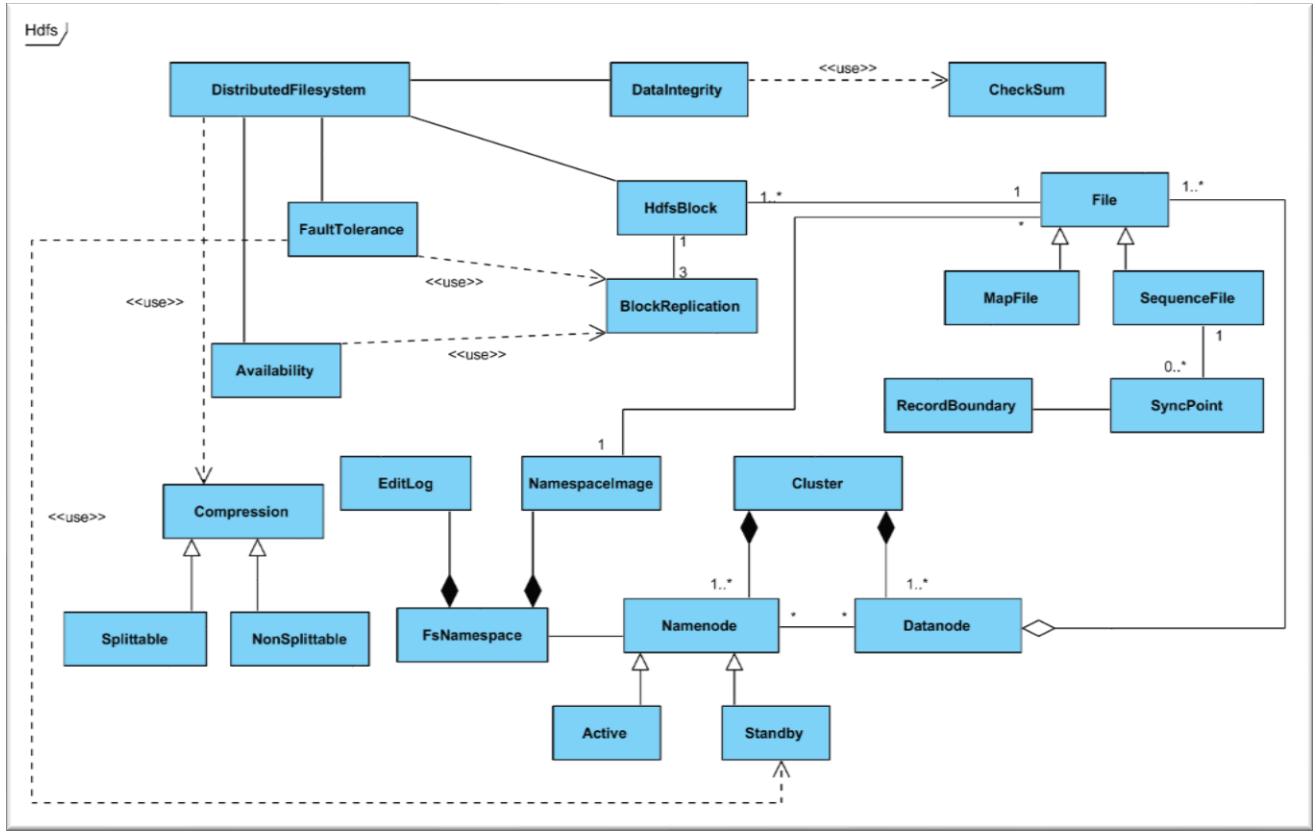


Figura 2.1 Diagramma delle classi kite-level raffigurante Hadoop Distributed Filesystem.

Il filesystem distribuito Hdfs assicura l'integrità dei dati (data integrity) e la disponibilità (availability) degli stessi ed è tollerante ai guasti (fault tolerance).

L'integrità dei dati è assicurata mediante l'utilizzo di un codice per il rilevamento degli errori (nello specifico una variante di CRC-32, nota con la sigla CRC-32C, è utilizzata). Infatti, ogni volta viene memorizzato un nuovo file, automaticamente viene calcolato e memorizzato il codice CRC-32C. Quando un file viene letto, in precedenza alla sua apertura, il sistema verifica l'integrità calcolando nuovamente il codice per il rilevamento dell'errore e confrontandolo con quello memorizzato. Nel caso in cui venga rilevato un errore, il client riceve un'eccezione (di tipo sottoclasse di IOException).

A livello Hdfs un cluster è composto da (tipicamente) una coppia di namenode, in configurazione active-standby, e da un insieme di datanode.

Un generico namenode gestisce il filesystem namespace (filesystem tree, metadati, ...) e memorizza queste informazioni su disco locale utilizzando due file: edit log (che contiene i metadati) e namespace image (che contiene il filesystem tree).

La configurazione active-standby dei namenode permette la tolleranza ai fallimenti, infatti, con una configurazione a singolo namenode, questo risulterebbe un single point of failure.

I datanode hanno il compito di memorizzare e recuperare blocchi su richiesta del client o del namenode attivo.

Un file è memorizzato in blocchi indipendenti, tipicamente da 128 mb. Questa astrazione permette al sistema di memorizzare file che eccedono la capacità di ogni disco fisico nel data center.

Per assicurare la tolleranza ai guasti, ogni blocco è replicato, di default tre volte, su macchine distinte. Inoltre la replicazione dei blocchi permette di aumentare il throughput del sistema; questo migliora la disponibilità.

In Hadoop i file sono di due tipi: sequence file e map file.

I sequence file forniscono una struttura dati persistente di tipo chiave-valore. Su questi file, in corrispondenza ad una chiava, è possibile inserire un sync point. Questi ultimi definiscono dei record boundary, ovvero degli spezzoni di file a cui si può fare riferimento durante la lettura. Inoltre, in MapReduce, i sync point permettono di processare ogni spezzone del file, in modo indipendente, da map task differenti.

I map file, a differenza dei sequence file, sono ordinati rispetto alle chiavi e queste sono indicizzate; permettono quindi l'accesso diretto ai record salvati.

In Hdfs è possibile comprimere i file. Diversi formati sono compatibili (gzip, lzo, snappy, ...). Alcuni di questi sono suddivisibili (“splittable”, ad esempio il formato lzo) mentre altri non lo sono (gzip, snappy). Nella prospettiva di

MapReduce conviene sempre utilizzare formati splittable i quali permettono di processare le diverse parti costituenti il file indipendentemente, da map task differenti. (Figura 2.1) [11]

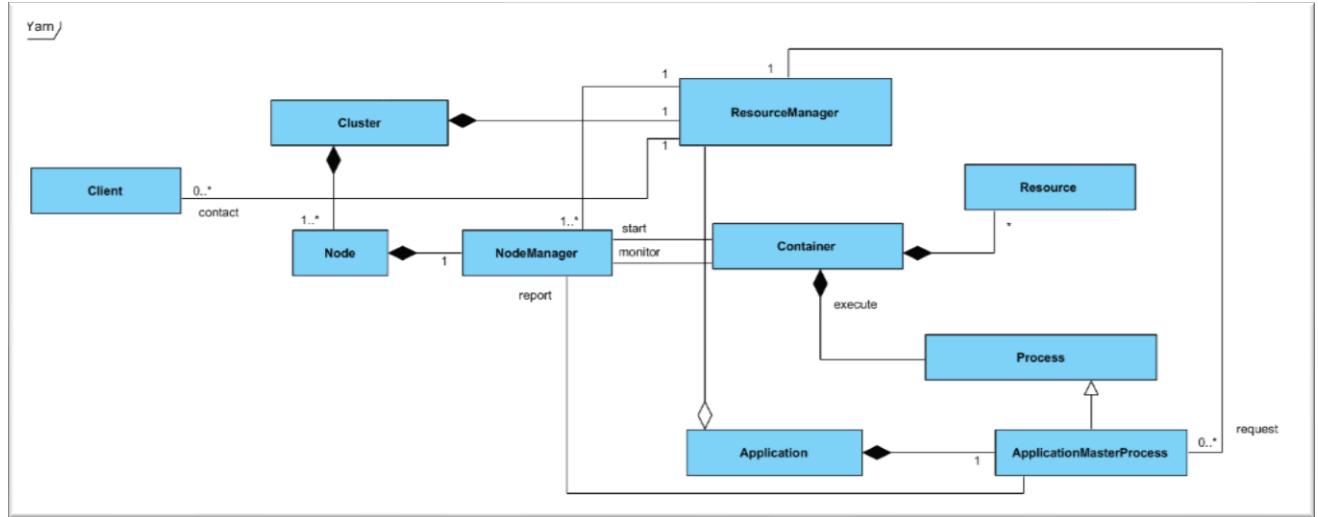


Figura 2.2 Diagramma delle classi kite-level raffigurante Hadoop Yarn.

Yarn ha il compito di allocare le risorse disponibili. Per farlo utilizza due tipi di processi demoni:

1. resource manager: uno per cluster; gestisce le risorse disponibili nel cluster;
2. node manager: uno per nodo; per eseguire e monitorare i container.

Un container è in grado di eseguire un processo specifico per una applicazione e associare a questo un certo quantitativo di risorse (CPU e storage).

Per eseguire una applicazione su YARN, il client contatta il resource manger e gli chiede di eseguire un application master process in un container.

Successivamente l'application master può richiedere al resource manager altri container per eseguire una computazione distribuita.

Il resource manager è in grado di lanciare nuovi container per mezzo dei node manager.

Una richiesta per un insieme di container può esprimere il quantitativo di risorse computazionali (CPU e memoria) necessarie per ognuno di essi e dei vincoli di località (un container su un nodo o rack specifico).

Le richieste possono avvenire up-front (tutte prima dell'esecuzione; i map task in MapReduce) o dinamicamente(i reducer task in MapReduce).

Tramite le richieste dinamiche, a livello application, è possibile gestire i task falliti eseguendoli nuovamente.

Nel caso di MapReduce ad una applicazione corrisponde un job utente. Quindi i container non vengono riutilizzati. Infatti è sempre presente l'overhead di creazione dei container. MapReduce 2 utilizza il resource manager per fare job scheduling; l'application master si occupa di monitorare i task (verificare lo stato, la percentuale completamento e rilanciare i task falliti).

Un processo timeline server viene eseguito e memorizza lo storico dei job completati.

I node manager inviano all'application master le informazioni per monitorare i task. (Figura 2.2) [11]

2.4 MapReduce

MapReduce è un modello che alterna la computazione sequenziale a quella parallela, sviluppato nella nota azienda di Mountain View, Google.

L'atomo dell'informazione, in MapReduce, è la coppia (chiave, valore). L'input per un qualsiasi algoritmo è un insieme di tali coppie. La computazione avviene in tre stage:

1. Map stage: il mapper m prende in input una singola coppia e produce, come output, un certo numero di nuove coppie. Input differenti possono essere eseguiti da macchine differenti, in parallelo.
2. Shuffle stage: mediante uno shuffler, l'output dei mapper viene riorganizzato, in modo tale che per ogni chiave distinta k venga generata una coppia (k, lista) dove la lista contiene tutti i valori associati alla chiave k . Idealmente, ognuna di queste nuove coppie viene inviata ad una macchina differente. In Hadoop MapReduce lo shuffler prende il nome di combiner.
3. Reduce stage: il reducer r prende in input una coppia $(\text{chiave}, \text{lista})$ per come precedentemente descritta e dà in output un multiset contenente coppie $(\text{chiave}, \text{valore})$ tutte aventi stessa chiave.

Si richiede che i mapper siano stateless, questo permette la computazione parallela.

Nel passo di riduzione (o Reduce stage) la computazione avviene in parallelo in quanto differenti istanze del reducer lavorano su coppie $(\text{chiave}, \text{lista})$ con chiave differente.

Un programma può essere composto da più round, ovvero è possibile reiterare il processo appena descritto. [12]

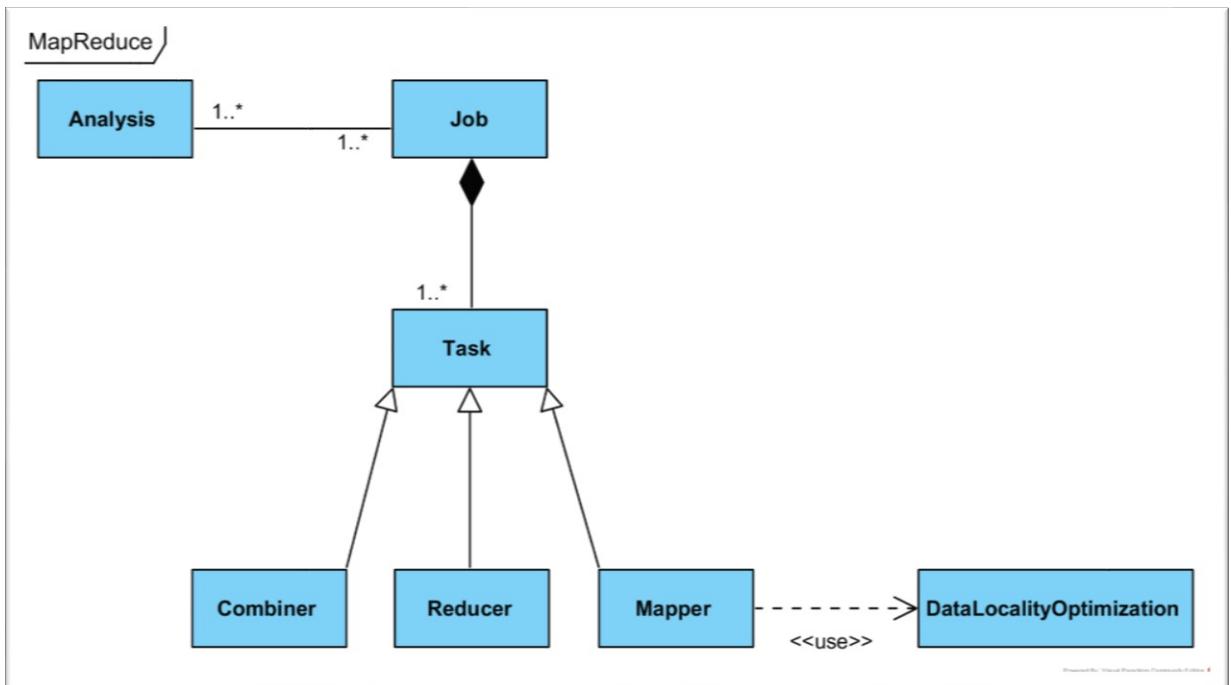


Figura 2.3 Diagramma delle classi kite-level raffigurante Hadoop MapReduce.

Per condurre un’ analisi, sfruttando il parallelismo fornito dalla implementazione Hadoop MapReduce, è necessario esprimere la nostra query come un MapReduce Job.

Il job suddiviso in task: map task e reducer task, schedulati da Yarn.

Hadoop partiziona l’input in input split e per ogni input split viene creata una istanza del tipo generico **Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT>** che processa ogni coppia (chiave, valore) nell’input split separatamente. Hadoop prova a creare questa istanza sul nodo (o macchina) del cluster dove i dati di input risiedono al fine di evitare il trasferimento dei dati di input (data locality optimization). L’output è scritto su disco locale.

L’input ad un singolo reduce task è normalmente l’output di tutti i mapper. Tuttavia è possibile configurare il job in modo tale che utilizzi più reducer. In questo caso, l’output dei mapper viene partizionato e ogni elemento della partizione è dato in input ad un reducer differente.

Hadoop permette all'utente di specificare come effettuare questo partizionamento, tramite un combiner. Questo viene eseguito su ogni output di ogni mapper, localmente. Ovvero i combiner implementano lo shuffle stage. (Figura 2.3) [11]

2.5 K-means con MapReduce

Con MapReduce è possibile implementare l'algoritmo K-means senza la pretesa che il data set risieda in memoria nella sua interezza. Ciò rende possibile processare una grande mole di dati.

Il data set di input viene preliminarmente salvato come un sequence file di coppie (chiave, valore), dove la chiave è un semplice offset in bytes dal punto di inizio del file; il valore è un record o punto nello spazio considerato. Analogamente, il file di input contenente i centri iniziali, viene salvato come sequence file.

Nell'algoritmo K-means, la parte che richiede più risorse in tempo dal punto di vista computazionale corrisponde al calcolo delle distanze. Infatti è necessario, al fine di associare ogni punto al centro più vicino, calcolare $n \cdot k$ distanze, dove n è la cardinalità del data set e k è il numero di cluster che si vogliono ottenere. Questo processo avviene per mezzo della funzione mapper.

Il mapper prende in input: una coppia (chiave, valore), proveniente dal sequence file relativo ai punti del data set sopra descritto; la variabile globale centers contente tutti i centri e restituisce una coppia (chiave', valore') dove chiave' è un riferimento al cluster a cui il punto valore' è assegnato. L'algoritmo è il seguente:

```

Costriisci il punto p n-dimensionale partendo da valore
distanzaMin = ∞
indice = -1
for i = 0 to length(centers)
    dist = distanza(p, centers[i])
    if dist < distanzaMin
        distanzaMin = dist
        riferimento = i
        chiave' = riferimento
        valore' = valore
    return (chiave', valore')

```

Agli output di ogni singolo mapper applichiamo una funzione combiner.

L'algoritmo prende in input una coppia (chiave, valore) dove la chiave è un riferimento ad un cluster e valore è una lista di punti tutti assegnati al cluster con riferimento chiave. In output restituisce una coppia (chiave', valore') dove chiave' è un riferimento ad un cluster e valore' è una stringa contenente sia la somma parziale che il numero di punti che hanno contribuito a tale somma.

Inizializza un array vuoto di punti V con stessa dimensionalità dei punti in valore
num = 0
for each v in valore:
 costruisci il punto p partendo da v
 V = V + p
 num = num + 1
 chiave' = chiave
 valore' è una stringa contenente V e num
return (chiave', valore')

La funzione reduce calcola la media di tutti i punti associati allo stesso riferimento di cluster.

La funzione prende in input una coppia (chiave, valore) dove chiave è un riferimento ad un cluster e valore è una lista di somme parziali di punti associati allo stesso cluster con riferimento chiave. In output l'algoritmo restituisce una coppia (chiave', valore') dove chiave' è un riferimento ad un cluster e valore' è un centroide per l'iterazione successiva. L'algoritmo è il seguente:

Inizializza un array vuoto di punti M con stessa dimensionalità dei punti in valore
num = 0

for each v,n in valore:

costruisci il punto p corrispondente alla n-pla in v

M = M + p

num = num + n

chiave' = chiave

valore' = M / num

return (chiave', valore')

Quanto descritto sino ad ora rappresenta un' iterazione dell'algoritmo K-means, è necessario di conseguenza reiterare il processo dando in input al mapper quanto emesso da tutti i reducer.

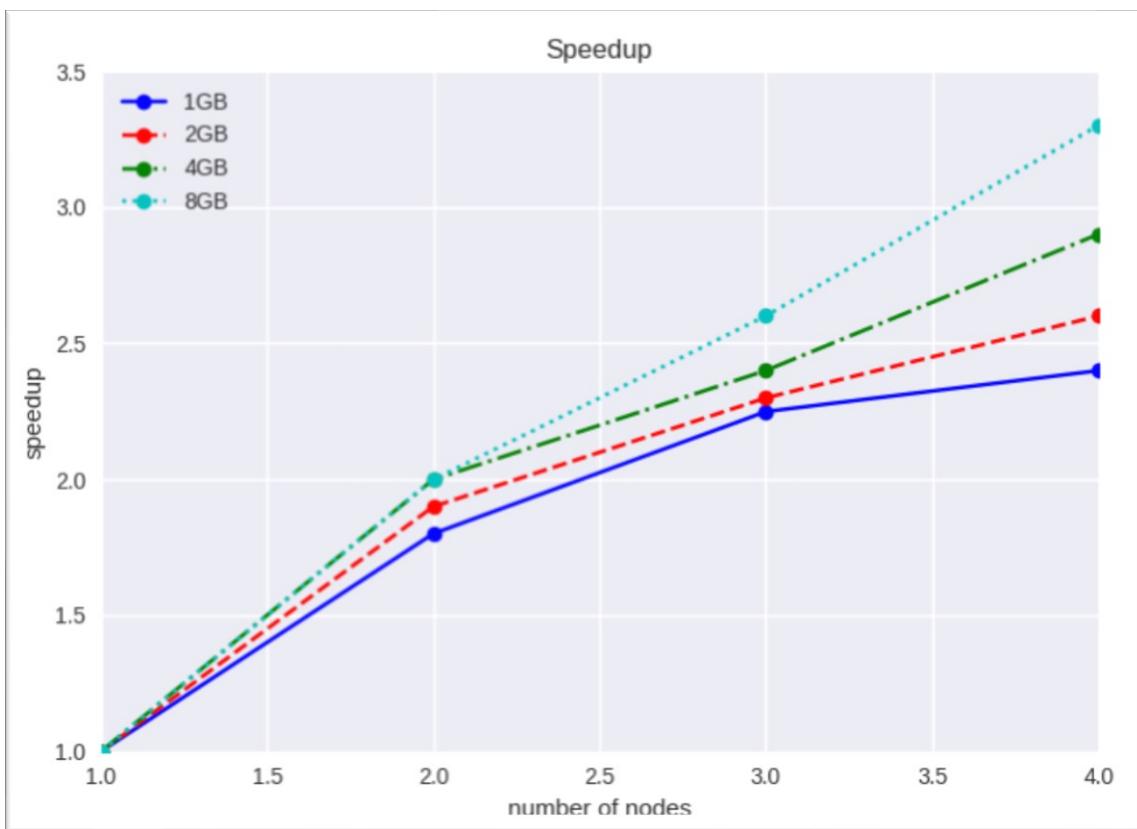


Figura 2.4 Speed up calcolata per numero di nodi pari a 1, 2, 3 e 4 con data set di dimensione 1GB, 2GB, 4GB e 8GB.

Possiamo interpretare la speed up come la capacità del sistema di aumentare le sue performance, per un fissato carico di lavoro, sfruttando un quantitativo maggiore di risorse.

Più precisamente,

$$\text{speed up} = \frac{\text{tempo necessario alla risoluzione del problema con poche macchine}}{\text{tempo necessario alla risoluzione del problema con tante macchine}}$$

Il sistema si comporta bene in quanto il risultato è approssimativamente lineare. (Figura 2.4)

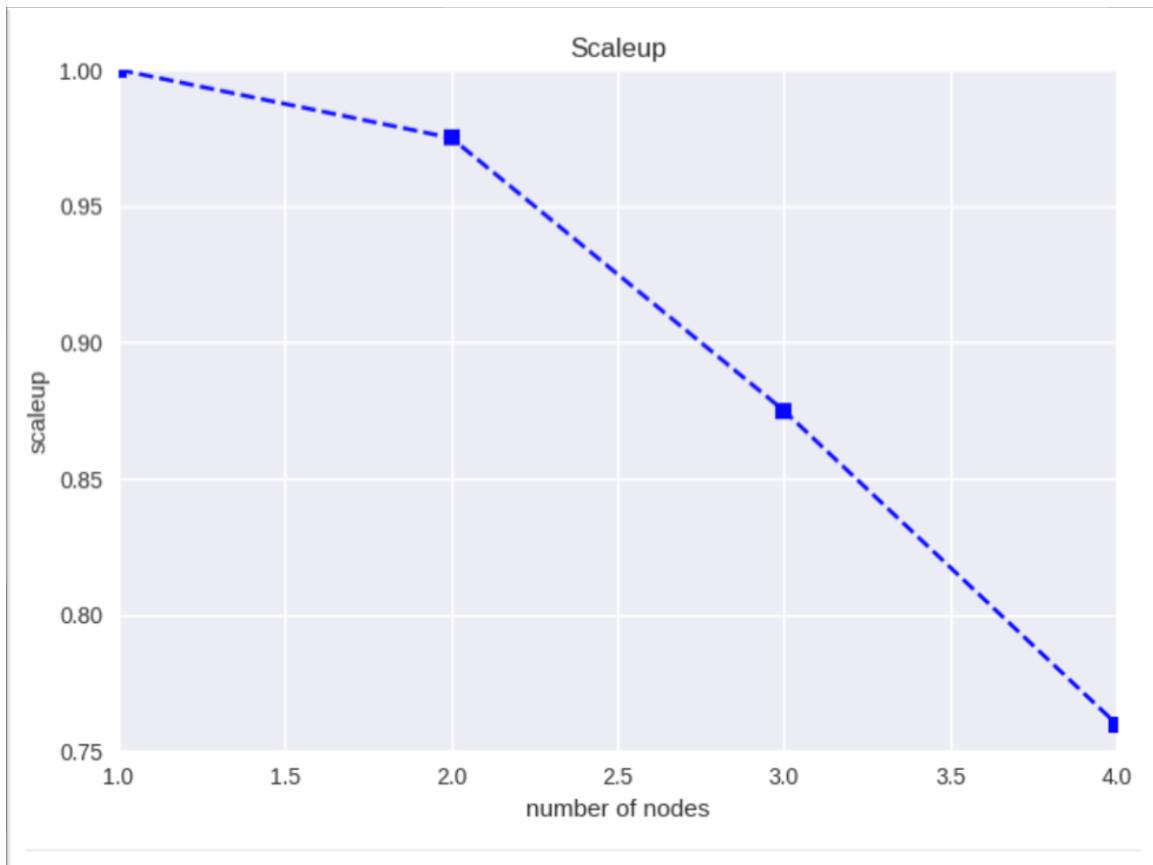


Figura 2.5 Scaleup calcolata per numero di nodi pari a 1,2,3 e 4.

Possiamo invece interpretare la scale up come l’abilità del sistema di mantenere lo stesso livello di performance quando sia il carico di lavoro (la cardinalità del data set) che la quantità di risorse disponibili (nodi che compongono il computer cluster) aumentano proporzionalmente.

Più precisamente,

scale up

$$= \frac{\text{tempo necessario alla risoluzione del problema con data set piccolo e poche macchine}}{\text{tempo necessario alla risoluzione del problema con data set grande e tante macchine}}$$

Anche in questo caso la scale up evidenzia delle ottime performance del sistema in quanto approssimativamente lineare. (Figura 2.5) [13]

Capitolo III

Valutazione Empirica

In questo capitolo si descrive una selezione di implementazioni K-Means in MapReduce. Le prestazioni degli algoritmi selezionati sono state valutate in termini di qualità dei cluster e tempi di computazione, considerando un data set reale e due data set artificiali.

3.1 Introduzione

Nel lavoro di tesi è stata effettuata un'analisi comparativa delle implementazioni K-Means su Hadoop MapReduce pubblicate su *GitHub*.

Le seguenti implementazioni sono state selezionate (per nickname utente):

- *himank*: questa implementazione è stata subito scartata in quanto effettua l'operazione di clustering considerando solo punti unidimensionali;
- *jgalilee*: questa implementazione è adatta agli scopi richiesti;
- *ics621hawaii*: in modo simile all'implementazione *himank*, questa permette solo una rappresentazione bidimensionale delle osservazioni e, non essendo specificata la versione di Hadoop utilizzata, non è stato possibile né circoscrivere né risolvere un problema riguardo la configurazione del job, anch'essa è stata quindi scartata;

- *jungblut*: questa implementazione è stata scartata in quanto utilizza la distanza L1 (o city block). Inoltre l'implementazione dà in output la rappresentazione in memoria dei centri finali piuttosto che la loro rappresentazione in caratteri interpretabili;
- *yhfjhf*: questa implementazione è adatta agli scopi richiesti;
- *cosc6339_s17*: questa implementazione, non reperibile su GitHub ma piuttosto all'indirizzo http://www2.cs.uh.edu/~gabriel/courses/cosc6339_s17, è stata scartata in quanto impone una rappresentazione bidimensionale delle osservazioni ed è in grado di costruire solo 2 cluster.
- *BrizziB*: questa implementazione è adatta agli scopi richiesti.

In definitiva le implementazioni scelte sono quelle degli utenti GitHub: *BrizziB* (<https://github.com/BrizziB>), *Jgalilee* (<https://github.com/jgalilee>) e *Yhfjhf* (<https://github.com/yhfjhf>).

3.2 Data set reale

Al fine di valutare le performance delle differenti implementazioni dell'algoritmo K-means è stato utilizzato il data set (reale) Indian Pines. (Figura 3.1)



Figura 3.1: Scena del data set Indian Pines

Indian Pines, catturato con il sensore AVIRIS, contiene 145*145 osservazioni, ognuna avente 200 bande spettrali (delle 245 acquisite sono state rimosse le bande che riguardano l'assorbimento dell'acqua). La scena raffigura per due terzi un terreno agricolo, la parte restante: una foresta, dell'altra vegetazione naturale e delle costruzioni artificiali. Tra queste ultime un'autostrada, una ferrovia ed alcuni edifici di vario tipo.

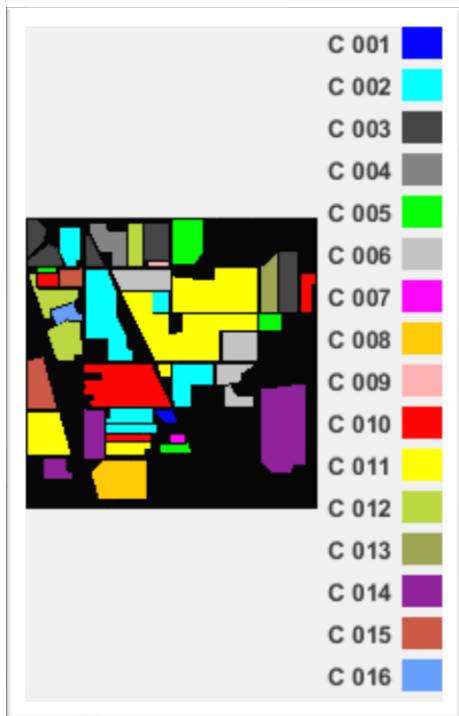


Figura 3.6: Ground Truth del data set Indian Pines.

Alla classe C002 corrisponde una piantagione di mais spontanea, alla classe C011 una piantagione di semi di soia semispontanea, alla C014 una foresta e alla c010 una piantagione di soia spontanea.

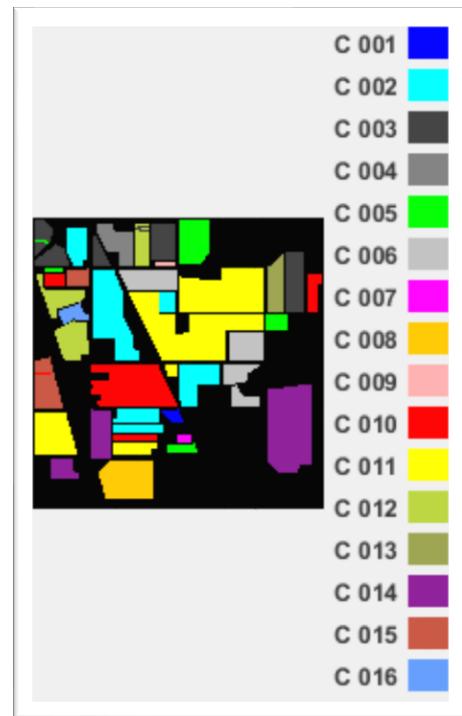


Figura 3.7: Ground Truth del data set modificato artificialmente derivato da Indian Pines.

Nella scena sono presenti 16 diverse classi (Figura 3.2), le più ricorrenti sono:

- piantagione spontanea di mais (1428 pixel)
- piantagione di soia semispontanea (2455 pixel)
- foresta (1265 pixel)
- piantagione di soia spontanea (972 pixel)

Da questo data set è stato derivato un data set multitemporale, fatto di due immagini; la prima corrisponde a quanto descritto precedentemente, la seconda è stata ottenuta

modificando manualmente i pixel nelle posizioni $x = 5, 52 < y < 59$; $x = 6, 51 < y < 55$; $x = 7, 53 < y < 59$; $x = 12, 0 < y < 8$; $x = 13, 6 < y < 9$; $x = 78, 0 < y < 10$. (Figura 3.3)

3.3 Data set artificiali

Al fine di valutare, con l'indice di Silhouette, e verificare, mediante l'indice di Rand, il corretto funzionamento delle implementazioni dell'algoritmo K-means sono stati costruiti artificialmente due data set. Entrambi contengono osservazioni fintizie rappresentate in uno spazio bidimensionale. Il data set 2d3c contiene 750 000 osservazioni, raggruppate in 3 cluster contenenti ognuno 250 000 punti. (Figura 3.4)

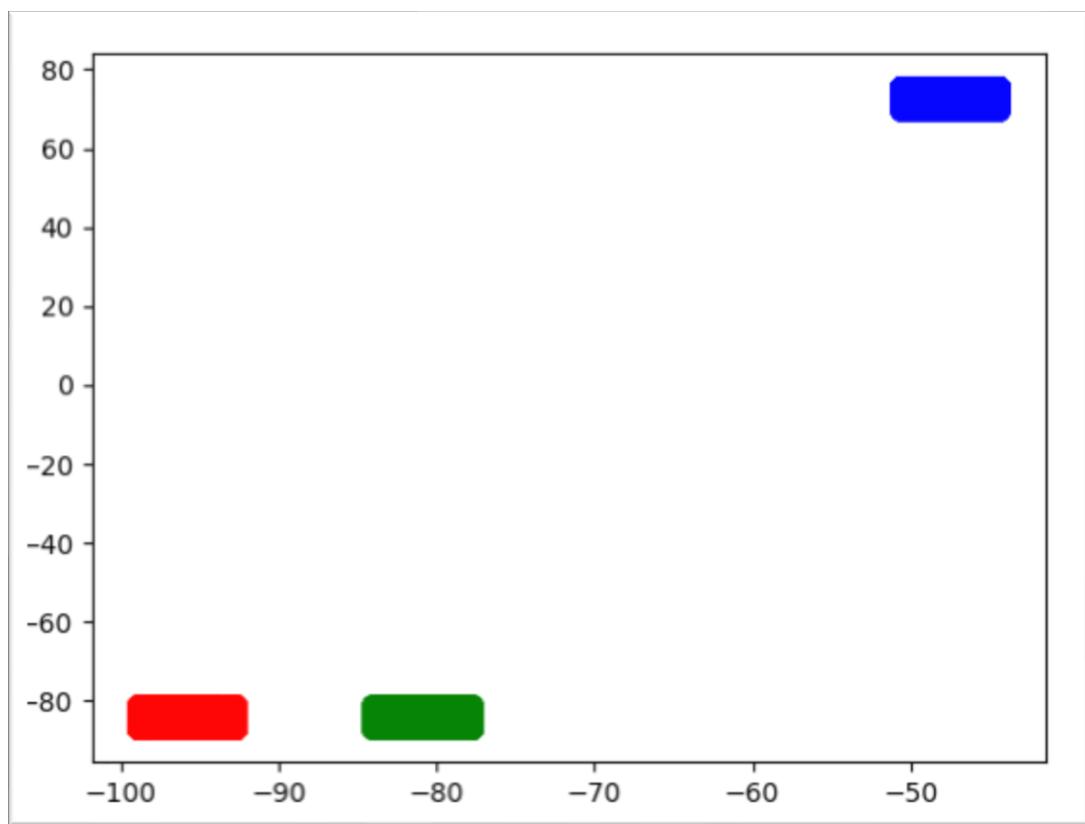


Figura 3.8: Diagramma a punti del data set 2d3c

Il data set 2d5c, invece, contiene 1 250 000 osservazioni, raggruppate in 5 cluster contenenti ognuno 250 000 punti. (Figura 3.5)

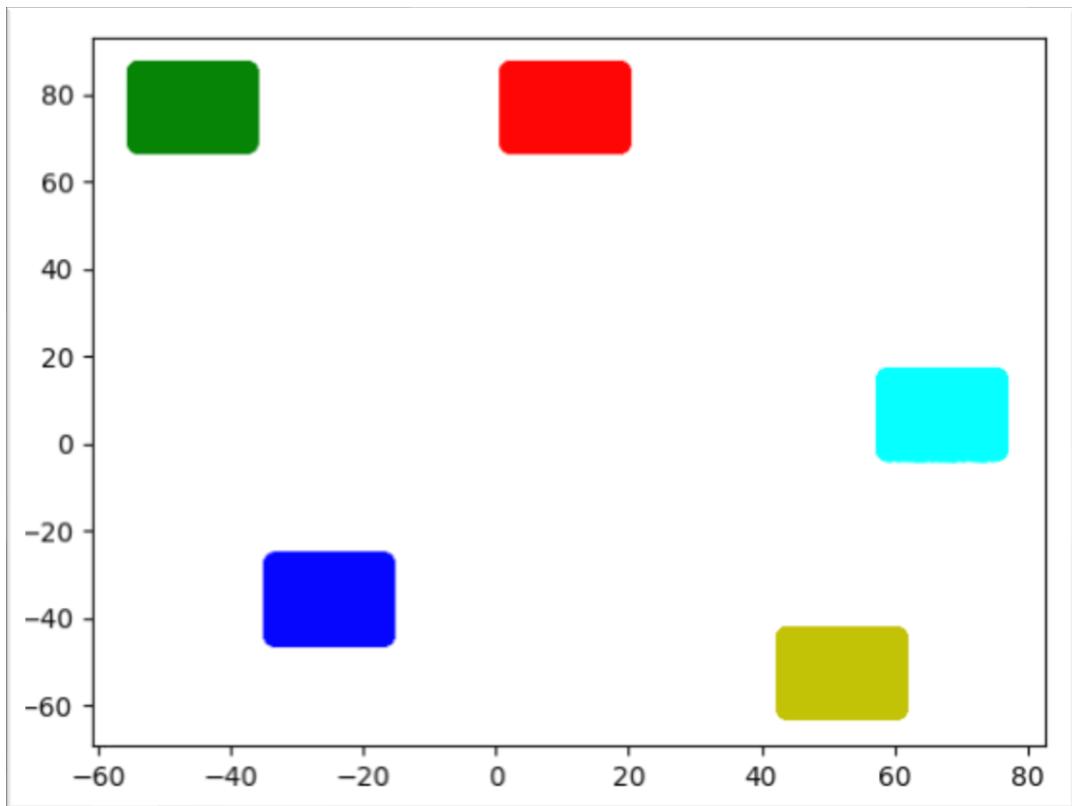


Figura 3.9: Diagramma a punti del data set 2d5c

La costruzione di questi data set è avvenuta utilizzando l'algoritmo GENERATE-CENTERS che definisce i rappresentanti di cluster:

GENERATE-CENTERS_{ORIGIN, BOUND}(dimensionality, quantity)

Genera un vettore *centers* di *quantity* vettori reali di dimensionalità *dimensionality* ...

... distinti scelti casualmente nell'intervallo...

...[*ORIGIN, BOUND*) \cup (*-BOUND, ORIGIN*]

return *centers*

I valori utilizzati per i parametri ORIGIN e BOUND dell'algoritmo sono 0 e 100 rispettivamente.

L'algoritmo GENERATE-EXAMPLE è stato utilizzato per generare i punti appartenenti ad ogni singolo cluster in modo tale che, punti appartenenti a cluster differenti, non si sovrappongano:

GENERATE-EXAMPLES(*center*, *quantity*, *dimensionality*, *window*)

Sia *result* una lista di vettori di dimensione *dimensionality*

for *i* = 0 **to** *quantity*

Sia *example* un vettore di dimensione *dimensionality*

for *j* = 0 **to** *dimensionality*

Genera un numero reale *x* casuale nell'intervallo...

...[0, *window*/2) U (-*window*/2, 0]

example[*j*] = *center*[*j*] + *x*

if *distance*(*center*, *example*) < *window*/2

result.add(*example*)

else

i = *i* - 1

La probabilità che la 7^a riga venga valutata come falsa aumenta in modo proporzionale alla dimensionalità del data set, infatti l'operazione *distance*, calcola il quadrato della distanza euclidea; una somma di quadrati di differenze.

Se la dimensionalità del data set è 2, consideriamo un quadrato avente area 1 ed un cerchio ad esso inscritto con diametro 1. La probabilità che venga eseguita la parte else del costrutto equivale alla zona del piano esterna al cerchio ed interna al quadrato. Questa è pari a $1 - 2\pi r^2$. Se la dimensionalità è 3, la probabilità è $1 - \frac{4}{3}\pi r^3 > 1 - \pi r^2$. Continuando a ragionare in questo modo possiamo verificare che all'aumentare della dimensionalità aumenta la probabilità che la

riga venga valutata come falsa, incrementando il tempo di esecuzione dell'algoritmo. Questa è una manifestazione della *curse of dimensionality*.

3.4 Analisi e Progettazione dei software utilizzati

Un data set artificiale è salvato in un file di testo seguendo il formato

-82.36446746055663,63.90273811666855

-81.70565428662857,67.91853145222007

...

dove le prime k righe, per un data set da k punti ed n cluster, fanno riferimento al primo cluster; le righe da k+1 a 2k fanno riferimento al secondo cluster e così via.

L'output dell'implementazione BrizziB è salvato su file nella forma

-80.914590830844 66.5111812711113

-80.32974400996152 63.67437888508667

...

dove alla prima riga è stampato il centroide, le successive sono i punti che fanno parte del cluster rappresentato dal centroide stesso. È possibile che un file contenga più di un cluster, può accadere quindi che alla riga k sia stampato un centroide e dalla k+2 in poi i punti che fanno riferimento ad esso. La riga k+1 sarà vuota (`\t\n`).

Riconosciamo la presenza di un nuovo cluster dall'assenza del carattere di tabulazione.

L'output dell'implementazione Jgalilee è salvato su file di testo, nella forma

1 -47.55515667108383 72.89179626563877

2 -80.91459083084465 66.5111812711107

3 -95.80069117827341 -84.07661573696988

...

Ad ogni riga corrisponde il centroide (o rappresentante) di un cluster preceduto dal suo codice identificativo (a partire da 1). Sarà quindi necessario considerare il file di input di testo contenente tutti i punti del data set, nel seguente formato

-81.2622959409307 65.84777093920698

-80.32974400996152 63.67437888508667

-83.16982883084457 69.21286832653041

...

E associare ognuno di essi al cluster corretto, al fine di ricostruire il data set risultante dall'operazione di clustering.

L'output dell'implementazione Yhfyhf è salvato su file di testo nel seguente formato

0-46.4452 71.7803

1-46.596603 73.60119

2-49.32123 70.934906

...

Dove il primo numero identifica il cluster di appartenenza (a partire da 0). I rappresentanti di questo cluster sono stampati a video seguendo il formato:

[-47.555946 72.89613, -80.918594 66.511505, -95.79116 -84.07881]

Questi vengono trasferiti su file di testo composto da una singola riga coincidente con il formato sopra mostrato.

È necessario considerare la ground truth del data set Indian Pines scaricabile all’indirizzo <http://www.di.uniba.it/~appice/software/S2TEC/index.htm>.

È inoltre necessario tenere conto che i punti, in tutti i casi visti, possono essere espressi in notazione scientifica.

Problema 1 (Silhouette)

Considerato un data set $\in \{\text{BrizziB}, \text{Jgalilee}, \text{Yhfyh}\}$ si vuole calcolare l’indice di Silhouette.

Soluzione

Per risolvere il problema è necessario ottenere una rappresentazione, per ogni punto del data set in analisi, nella forma:

-80.32974400996152,63.67437888508667,1

Dove i primi due numeri reali, sono il punto bidimensionale considerato o, nel caso del data set Indian Pines tutti gli attributi spettrali e non del punto, seguita da un numero intero che corrisponde al codice identificativo del cluster di appartenenza del punto nell’output dell’implementazione scelta.

In definitiva, la soluzione, per una istanza del problema, consiste dei seguenti passi:

1. Il caricamento in memoria del data set considerato, memorizzando gli attributi di tutti gli esempi in una matrice di numeri reali e il codice identificativo di tutti gli esempi in un vettore di interi; in modo tale che nella posizione i della matrice troviamo gli attributi dell’ i -esima osservazione e nella posizione i del vettore troviamo il codice identificativo del cluster di appartenenza della stessa osservazione.
2. Il calcolo dell’indice di Silhouette partendo dalla matrice e dal vettore ottenuti al passo 1.

L'algoritmo SILHOUETTE utilizzato per il calcolo dell'indice di Silhouette prende in input la matrice A degli attributi e la il vettore C dei codici identificativi dei cluster di appartenenza, ottenuti al punto 1:

SILHOUETTE(A,C)

silhouette = 0

for i = 1 **to** rowlength(A)

 e = A[i]

 eWithin = 0

 eWithinNum = -1

 Sia eBetween un array di dimensione |distinct(C)|

 Sia eBetweenNum un array di dimensione |distinct(C)|

for j = 1 to rowlength(A)

 e2 = A[j]

if C[i] == C[j]

 eWithin = eWithin + distance(e, e2)

 eWithinNum = eWithinNum + 1

else

 eBetween[C[j]] = eBetween + distance(e, e2)

 eBetweenNum[C[j]] = eBetweenNum[C[j]] + 1

 eWithin = eWithin / eWithinNum

for k = 1 **to** length(eBetween)

 eBetween[k] = eBetween[k] / eBetweenNum[k]

 minBetween = min(eBetween)

 silhouette = silhouette + (minBetween – eWithin) / max(minBetween, eWithin)

return silhouette / rowlength(A)

La complessità in tempo dell'algoritmo è $O(n^2)$ dove n corrisponde al numero di punti nel data set. Tale complessità risulta subottimale in quanto trascura la

simmetria della funzione distanza; tuttavia la scelta di tale algoritmo deriva dalla necessità di ottenere una complessità in spazio tale da poter processare data set di numerosità superiore ad 1 milione di punti utilizzando una macchina di tipo consumer.

A seguito della precedente analisi è stato progettato ed implementato il software SilhouetteCalculator per il calcolo della Silhouette. Come diagramma delle classi UML ad alto livello (o kite level) consideriamo il seguente:

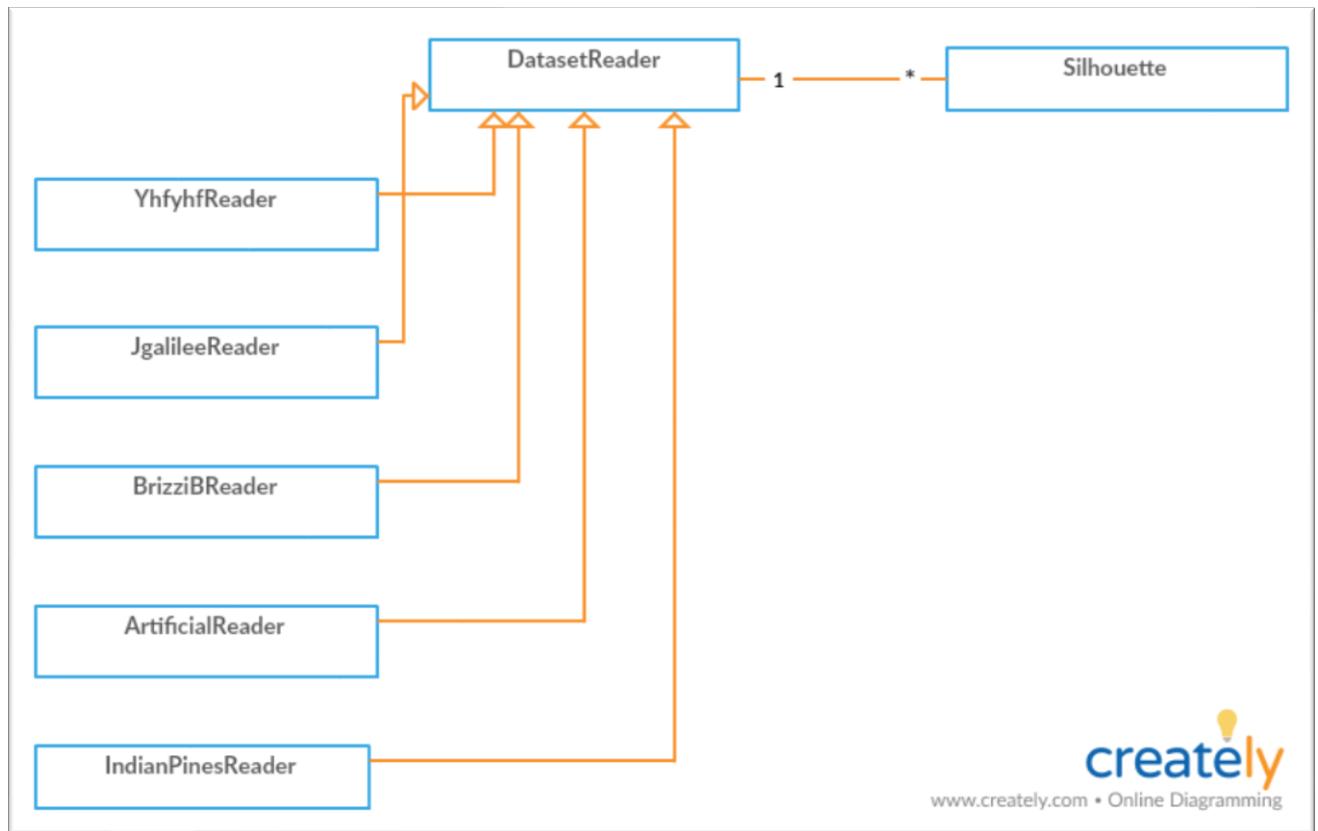


Figura 3.6: Diagramma delle classi kite-level del programma SilhouetteCalculator presente sul CD-ROM

dove le responsabilità sono così distribuite: (Figura 3.6)

- `DatasetReader` legge i punti da file assumendo che tutte le componenti, di ogni punto, siano coinvolte nell'operazione di clustering;
- `YhfyhfReader` legge il data set ottenuto dall'operazione di clustering effettuata dall'implementazione `Yhfyhf`;

- JgalileeReader legge il data set ottenuto dall'operazione di clustering effettuata dall'implementazione Jgalilee.
- BrizziBReader legge il data set ottenuto dall'operazione di clustering effettuata dall'implementazione BrizziB.
- ArtificialReader legge i dataset artificiali bidimensionali generati dal programma DatasetGenerator.
- IndianPinesReader: legge il data set Indian Pines per come presentato nel pacchetto del software S2Tec.
- Silhouette calcola l'indice di Silhouette per mezzo dell'algoritmo SILHOUETTE.

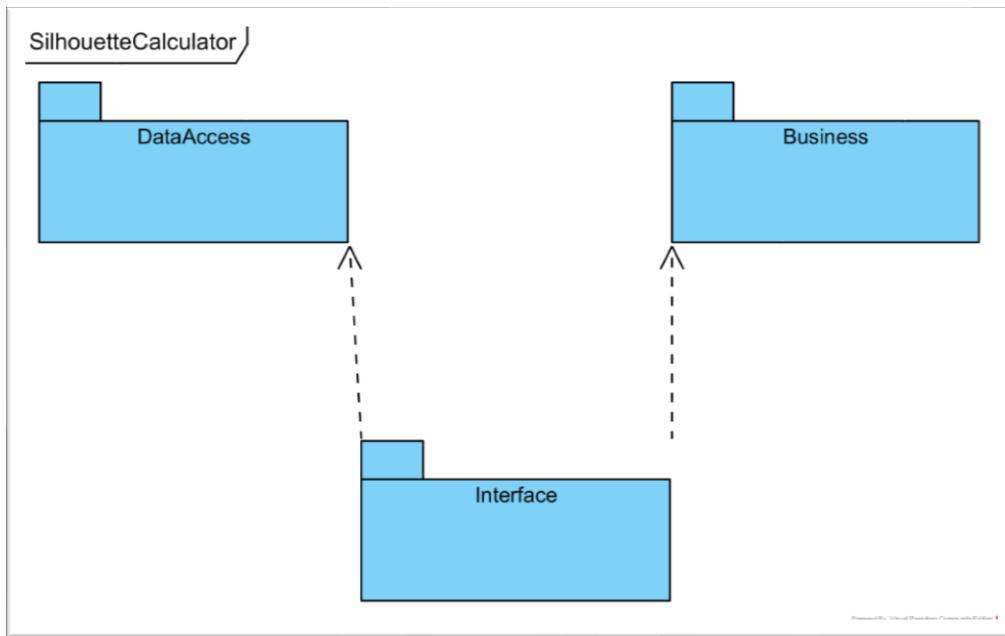


Figura 3.7: Diagramma dei package di *SilhouetteCalculator*

Al termine della prima iterazione, seguendo un approccio risk and responsibility driven, dopo aver associato al caso d'uso “calcolare la silhouette di 2d5c per Jgalilee” il rischio maggiore, il codice è stato strutturato utilizzando i seguenti package: (Figura 3.7)

- DataAccess: contiene le classi utili alla lettura dei data set;
- Business: contiene le classi utili al calcolo della Silhouette;

- Interface: contiene le classi che implementano l’interfaccia utente del programma.

Nel dettaglio, il package `DataAccess` è composto come segue:

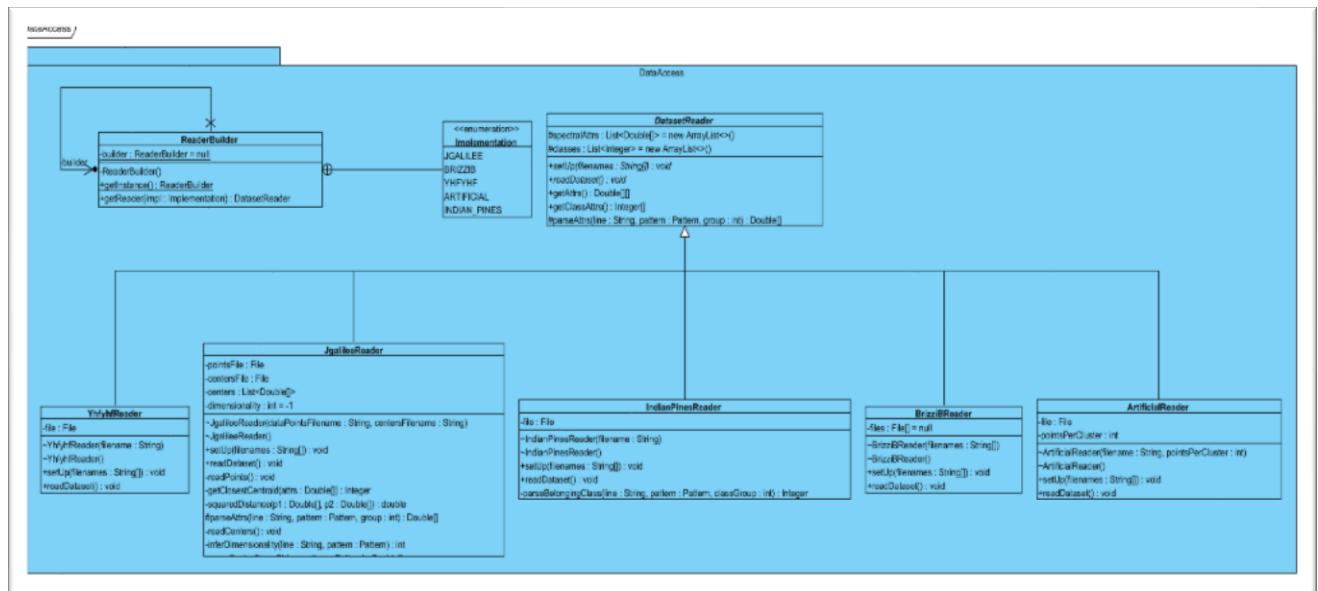


Figura 3.8: Diagramma delle classi fish-level del package `DataAccess` del programma `SilhouetteCalculator`

La classe astratta `DatasetReader` modella un generico data access object, astraendo le differenze nel formato dei record (o punti) presenti nelle varie implementazioni e dando la possibilità ai client di disinteressarsi di queste differenze. È possibile mediante gli oggetti concreti (`.+Reader`) leggere i dati delle tre implementazioni considerate o caricare in memoria i data set artificiali e Indian Pines.

Questa classe astratta specifica l'interfaccia delle sue sottoclassi mediante i metodi astratti `setUp`, utile ai client per configurare gli oggetti (`Reader`) ottenuti, e `readDataset` che carica il data set in memoria.

Inoltre la classe contiene conoscenza riguardo il parsing degli attributi spettrali di una singola riga del data set testuale considerato. Questa conoscenza è espressa mediante il factory method `parseAttrs`.

DatasetReader rappresenta un data set utilizzando una lista, implementata con array, contenente gli attributi spettrali di ogni osservazione presente nel data set secondo l'ordine in cui esse compaiono.

Inoltre è utilizzato una lista, anch'essa implementata mediante array contenente l'identificativo di classe, o di cluster, delle osservazioni, nell'ordine in cui esse compaiono nel data set.

Si è scelta tale implementazione per la lista in quanto non sono necessarie le operazioni di cancellazione.

La sottoclasse JgalileeReader mantiene un riferimento al file contenente il data set (pointsFile), un riferimento al file contenente i centroidi risultanti dall'operazione di clustering (centersFile) e un intero che memorizza la dimensionalità delle osservazioni (dimensionality). I primi due sono inizializzati dal costruttore parametrico o dal metodo setUp, che prendono in input i percorsi dei rispettivi file. La dimensionalità è calcolata durante la lettura.

La lettura del data set avviene per mezzo dei metodi readCenters, che memorizza i centroidi finali in una lista di vettori di reali a doppia precisione (centers), e readPoints che memorizza i punti.

Considerando che l'implementazione Jgalilee non restituisce un data set organizzato per cluster, piuttosto solo i rappresentanti di quest'ultimi, è necessario associare ogni osservazione al rappresentante più vicino. I metodi getClosestCentroid e squaredDistance permettono di eseguire questa operazione essendo in grado di restituire il centroide finale più vicino, data una osservazione, dopo averne calcolato la distanza.

Le altre sottoclassi di DatasetReader hanno un funzionamento simile, tenendo conto che non è necessario associare nuovamente le osservazioni ai rappresentanti di cluster ottenuti dall'operazione di clustering.

La classe ReaderBuilder è un singleton che astrae il processo di costruzione delle sottoclassi concrete di DatasetReader. La classe enumerativa Implementation, interna a ReaderBuilder, definisce le possibili implementazioni supportate da DatasetReader. La classe ReaderBuilder utilizza una istanza di Implementation, nel metodo getReader, per determinare il corretto DatasetReader da restituire. (Figura 3.8)

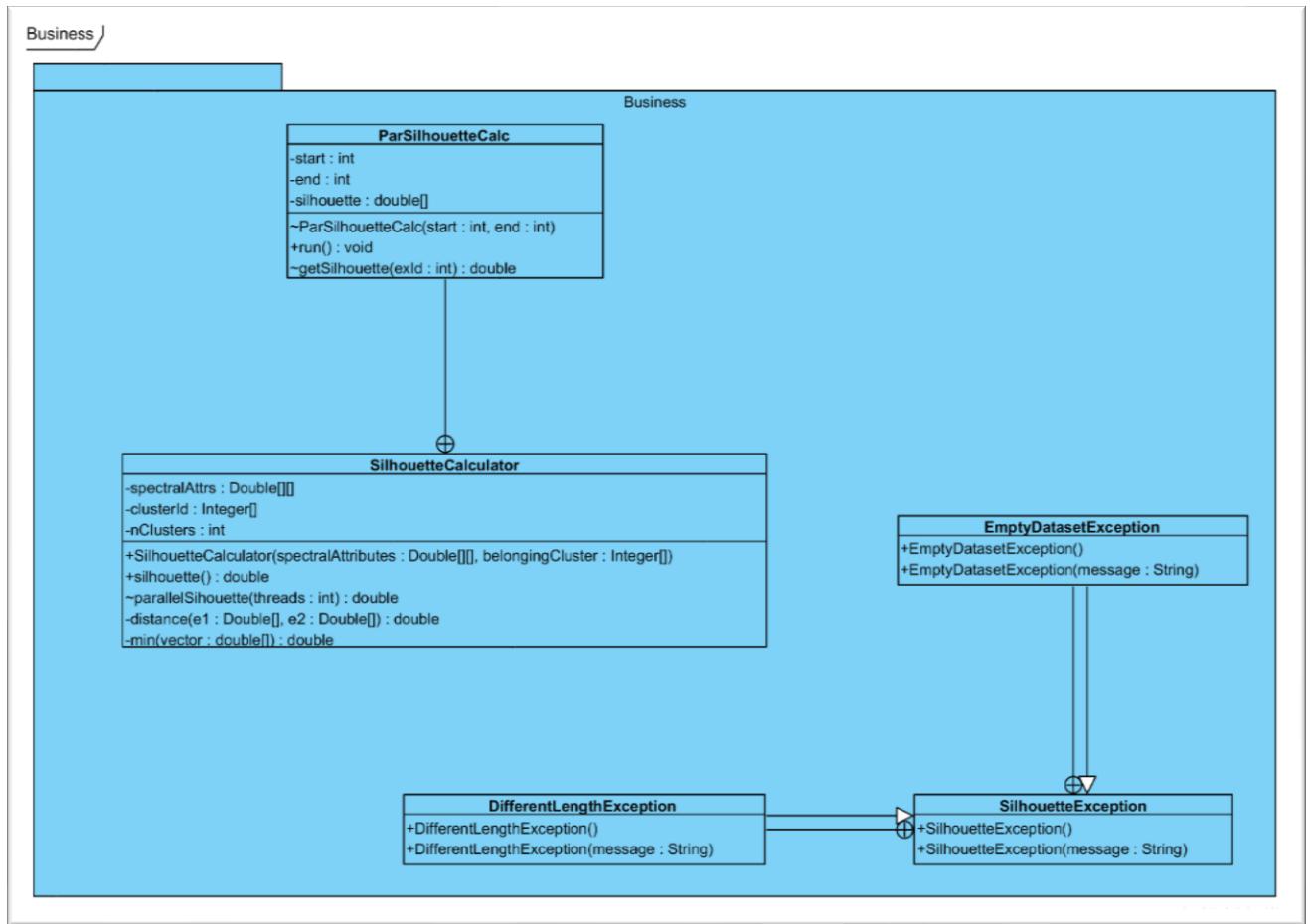


Figura 3.9: Diagramma delle classi fish-level del package Business di SilhouetteCalculator

La classe SilhouetteCalculator implementa l'algoritmo per il calcolo della Silhouette (metodo `silhouette`). L'oggetto è costruito fornendo una matrice di attributi spettrali e il vettore “simmetrico” dei cluster di appartenenza, di ogni osservazione.

La inner class privata ParSilhouetteCalc gestisce il calcolo nel caso in cui questo venga eseguito in parallelo per mezzo del metodo SilhouetteCalculator.parallelSilhouette.

Quest'ultima nasconde gli indici della parte di data set considerata (start ed end), infatti il parallelismo utilizzato è di tipo dati.

Inoltre questa classe memorizza i valori di Silhouette per ogni osservazione nella parte considerata in un vettore di reali a precisione doppia.

SilhouetteException modella una generica eccezione nel package business. Le sottoclassi SilhouetteException.EmptyDatasetException e SilhouetteException.DifferentLengthException modellano rispettivamente l'eventualità di un data set vuoto e di un data set contenente osservazioni rappresentate in spazi metrici differenti. (Figura 3.9)

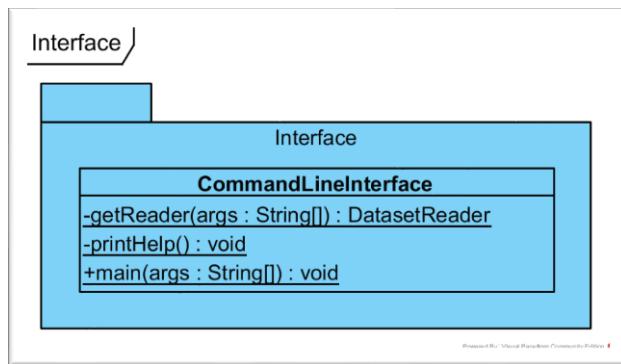


Figura.3 10: Diagramma delle classi fish-level del package Interface di SilhouetteCalculator

Il package Interface contiene una sola classe, la cui responsabilità è quella di implementare un'interfaccia a riga di comando per il software SilhouetteCalculator. Questa è implementata facendo uso di due metodi di classe: il metodo main è l'entry point del programma e gestisce, facendo uso dei metodi getReader e printHelp, tutta la logica relativa all'interfaccia. (Figura 3.10)

Problema 2 (Indice di Rand)

Considerata la coppia (data set artificiale, implementazione) con implementazione $\in \{\text{BrizziB}, \text{Jgalilee}, \text{Yhfyh}\}$, data set artificiale $\in \{\text{Indian Pines}, \text{2d3c}, \text{2d5c}\}$, si vuole calcolare l'indice di Rand.

Soluzione

Per risolvere il problema è necessario ottenere una rappresentazione, per ogni punto del data set in analisi, nella forma:

-80.32974400996152,63.67437888508667,1,2

Dove i primi due numeri reali, sono il punto bidimensionale considerato o, nel caso del data set Indian Pines, la posizione x, y del pixel nell'immagine e i due valori interi successivi sono rispettivamente: il codice identificativo della classe di appartenenza del punto nel data set e il codice identificativo del cluster di appartenenza.

In definitiva, la soluzione, per l'istanza del problema (d,i), consiste dei seguenti passi:

1. Il caricamento in memoria del data set d e del data set i come insieme di coppie chiave-valore dove, considerato un punto, la chiave è univoca ed è una stringa contenente la posizione x, y del pixel. Il valore è il codice identificativo della classe, per il data set d e il codice identificativo del cluster, per il data set i, di appartenenza del punto.
2. La fusione dei due data set d ed i in un unico data set rappresentato da una matrice bidimensionale, le cui righe rappresentano esempi, le cui colonne sono la classe ed il cluster di appartenenza del punto.
3. Il calcolo dell'indice di rand partendo dal data set risultante dall'operazione al punto 2.

L'algoritmo RAND-INDEX utilizzato per il calcolo dell'indice di Rand prende in input la matrice M calcolata al punto 2:

RAND-INDEX(M)

a,d = 0

for i = 1 **to** M.length

 class1 = M[i][1]

 cluster1 = M[i][2]

for j = 1 **to** i-1

 class2 = M[j][1]

 cluster2 = M[j][2]

if class1 == cluster1 **and** class2 == cluster2

 a = a + 1

else if class1 != cluster1 **and** class2 != cluster2

 d = d + 1

return (a + d) / ((length(M) * (length(M) - 1)) / 2)

La complessità in tempo dell'algoritmo è $O(n^2)$, ciò è facilmente deducibile dal fatto che, i due cicli for iterano sull'insieme di tutte le possibili combinazioni degli n (pari al numero di osservazioni o punti) indici, senza ripetizioni. La cardinalità di questo insieme è $n*(n-1)/2$, ovvero $n^2/2 - n/2 = O(n^2)$.

Il programma RandIndexCalculator è stato progettato ed implementato al fine di risolvere concretamente il problema descritto.

Come diagramma delle classi concettuali consideriamo il seguente:

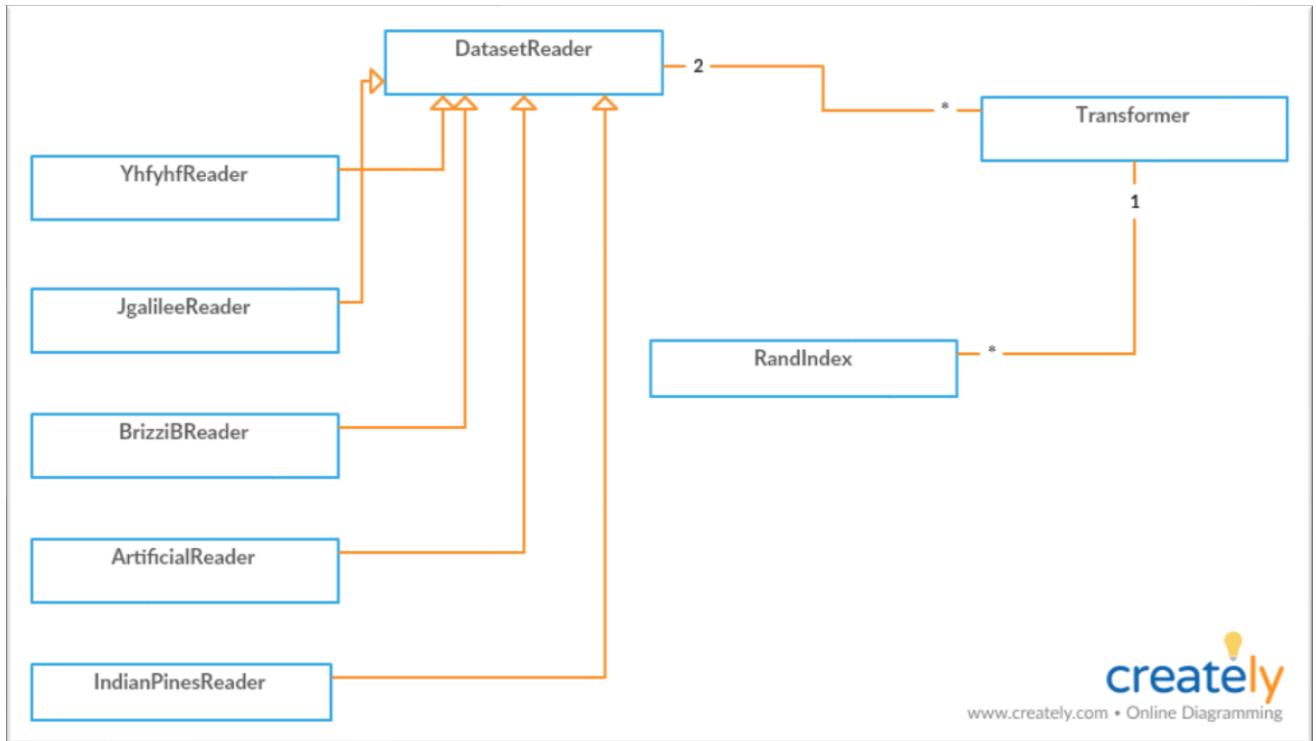


Figura 3.11: Diagrammi delle classi kite-level del programma RandIndexCalculator presente sul CD-ROM

dove le responsabilità sono così distribuite: (Figura 3.11)

- DatasetReader legge i punti da file assumendo che le prime due componenti di ognuno di essi identificano la posizione del punto nell’immagine;
- YhfyhfReader legge il data set ottenuto dall’operazione di clustering effettuata dall’implementazione Yhfyh;
- JgalileeReader legge il data set ottenuto dall’operazione di clustering effettuata dall’implementazione Jgalilee;
- BrizziBReader legge il data set ottenuto dall’operazione di clustering effettuata dall’implementazione BrizziB;
- ArtificialReader legge i dataset artificiali bidimensionali generati dal programma DatasetGenerator;
- IndianPinesReader: legge il data set Indian Pines per come presentato nel pacchetto del software S2Tec;
- Transformer fonde due data set contenti gli stessi punti in un unico data set;

- RandIndex calcola l'indice di Rand per mezzo dell'algoritmo RAND-INDEX.

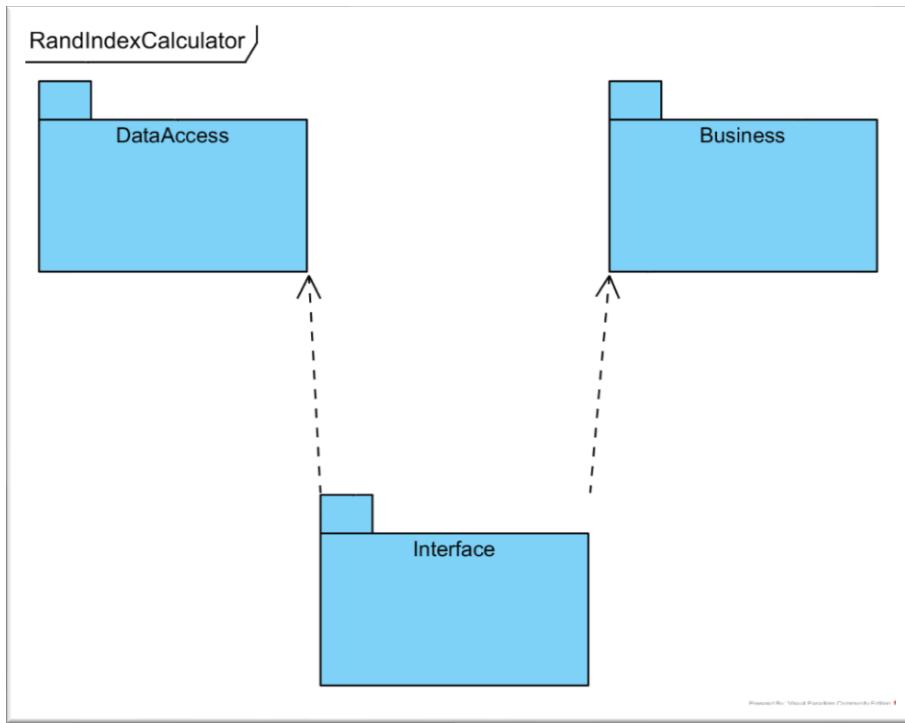


Figura 3.12: Diagramma dei package del programma RandIndexCalculator

Anche in questo caso, a seguito della prima iterazione, il programma è stato strutturato utilizzando i seguenti package: (Figura 3.12)

- `DataAccess`: contiene le classi utili alla lettura dei data set;
- `Business`: contiene le classi utili al calcolo dell'indice di Rand;
- `Interface`: contiene le classi che implementano l'interfaccia utente del programma.

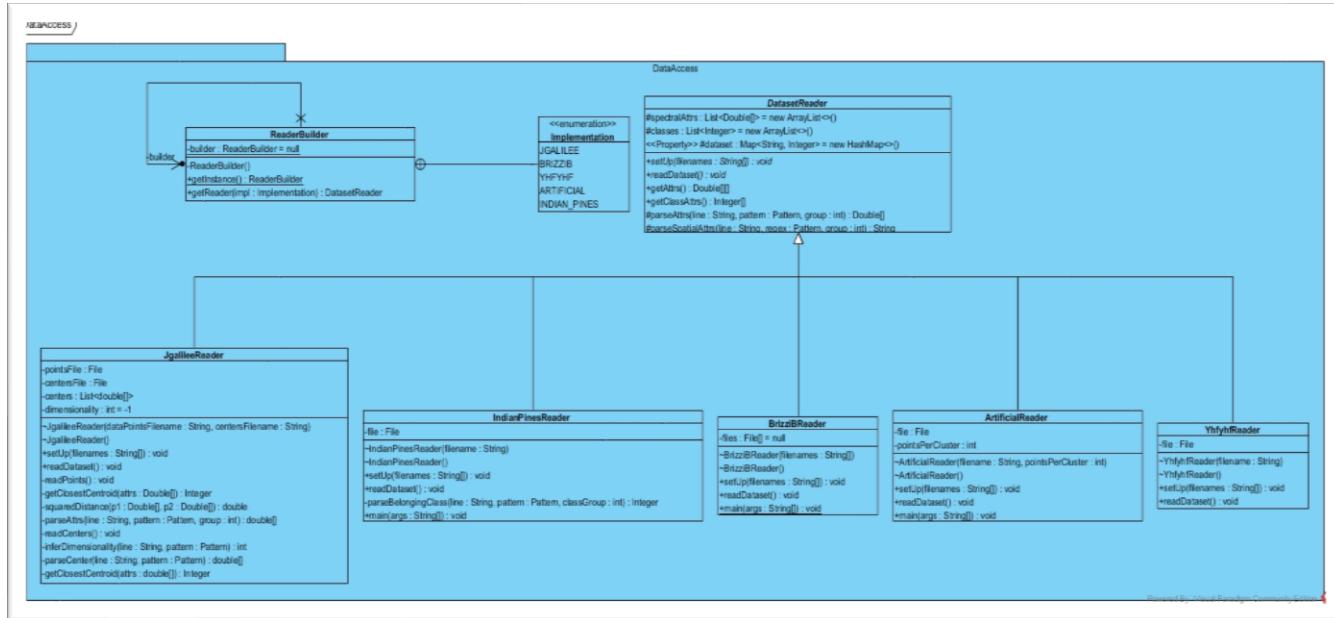


Figura 3.13: Diagramma delle classi del package DataAccess del programma RandIndexCalculator

La differenza rispetto al package DataAccess del programma SilhouetteCalculator risiede nella rappresentazione in memoria del data set. In questo caso esso è memorizzato utilizzando un contenitore associativo, nello specifico un'implementazione hash è stata scelta per garantire l'accesso (pseudo)diretto. (Figura 3.13)

Date le analogie fra i package DataAccess dei due programmi sino ad ora considerati, si sta valutando l'idea di fondere i due programmi durante la prossima iterazione.

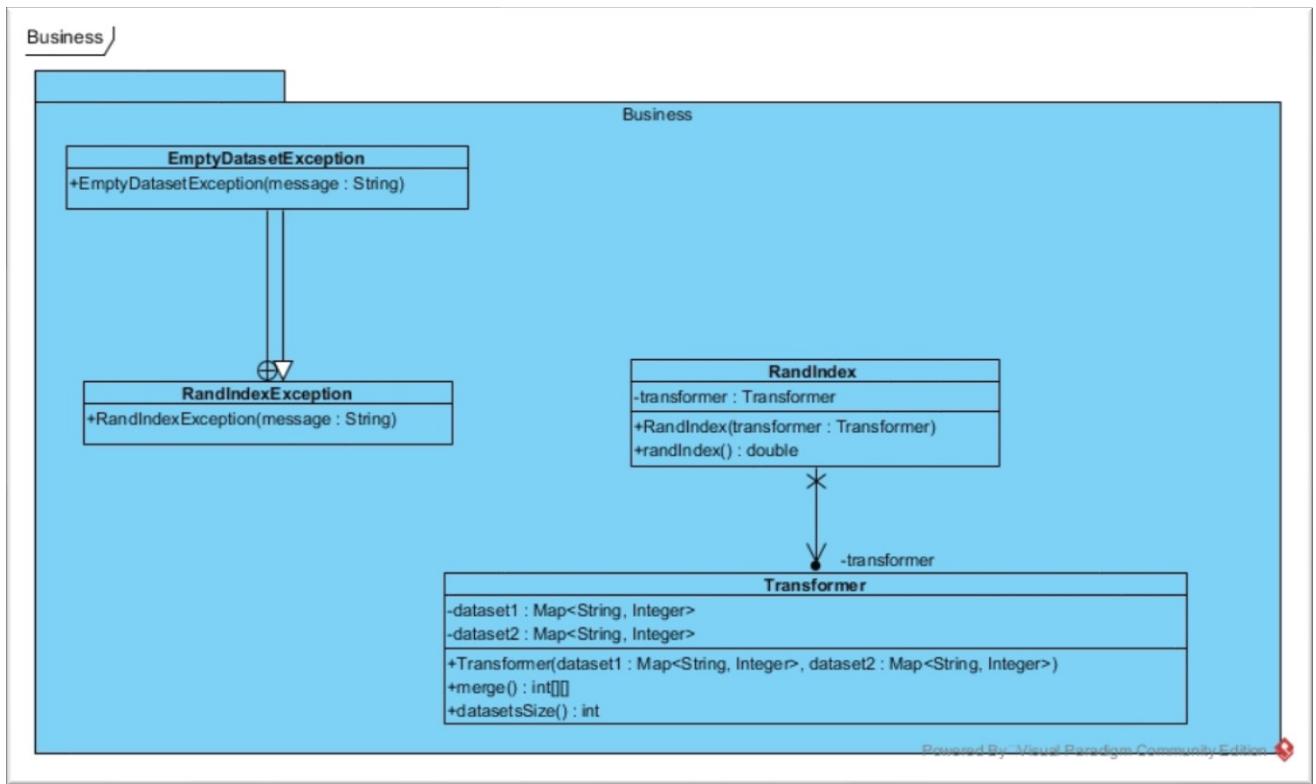


Figura 3.14: Diagramma delle classi del package Business di RandIndexCalculator

La classe RandIndex implementa l'algoritmo per il calcolo dell'indice di Rand facendo uso di un'istanza della classe Transformer. Quest'ultima fonde il data set annotato e lo stesso data set ottenuto dall'operazione di clustering in un unico data set, utilizzando il metodo merge, in una rappresentazione (chiave, valore) dove la chiave è frutto di un'operazione di hashing della stringa ottenuta dalla concatenazione delle variabili x, y dell'osservazione considerata. Il valore è a sua volta una coppia contenente classe e cluster di appartenenza dell'osservazione. (Figura 3.14)

RandIndexException modella una generica eccezione che può verificarsi durante la computazione effettuata dagli elementi di modello responsabili del calcolo dell'indice di Rand.

La sua sottoclasse EmptyDatasetException modella il caso in cui l'utente cerchi di calcolare l'indice di Rand di un data set vuoto.

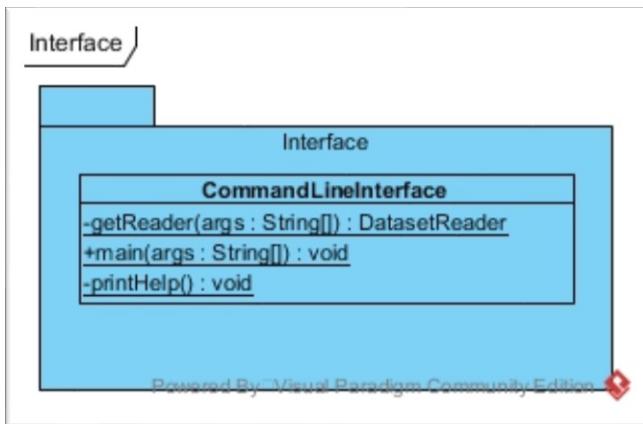


Figura 3.15: Diagramma delle classi del package Interface di RandIndexCalculator

Il package Interface contiene una sola classe che implementa un’interfaccia a riga di comando per il programma RandIndexCalculator. Nel particolare, ciò avviene mediante l’entry point e metodo di classe main, che sfrutta la conoscenza contenuta nei metodi di classe getReader (ha la responsabilità di restituire il reader corretto) e printHelp (ha la responsabilità di mostrare all’utente il messaggio di aiuto). (Figura 3.15)

3.5 Risultati e conclusione

La piattaforma utilizzata presenta le seguenti caratteristiche: Intel Core i7-8550U Quad Core, 8GB di memoria RAM, 2 SSD da 128GB e 256GB, scheda video AMD Radeon 530 con 4GB di memoria e sistema operativo Ubuntu 17.10.

Sono presentati in seguito i risultati ottenuti in cui, il tempo di esecuzione è calcolato come:

$$\frac{t_{fine} - t_{inizio}}{10^6} \text{ sec}$$

Nel calcolo dello stesso non è considerato il tempo necessario alla stampa su file dei punti di input, nelle implementazioni che lo prevedono.

Indian Pines	Silhouette	Indice di Rand	Tempo di esecuzione
BrizziB	0.270012237392783	0.854739031678725	1348101
Jgalilee	0.270012237392782	0.854739031678725	222610
Yhfyhf	0.243601232001879	0.857463617287846	13664

Tabella 3.10: Risultati dei test effettuati sul data set Indian Pines. L'implementazione Yhfyhf risulta la meno accurata rispetto alla Silhouette ma anche la più efficiente.

2d3c	Silhouette	Indice di Rand	Tempo di esecuzione
BrizziB	0.927449136768952	0.999992000067556	22179
Jgalilee	0.927449136768973	0.999992000067556	15088
Yhfyhf	0.927449136648963	0.999992000067556	16814

Tabella 3.2: Risultati dei test effettuati sul data set 2d3c. L'implementazione Jgalilee risulta essere la più efficiente.

2d5c	Silhouette	Indice di Rand	Tempo di esecuzione
BrizziB	0.867585181871393	0.9999920001152	28957
Jgalilee	0.867557953296446	0.9999920001152	17696
Yhfyhf	0.867583566139415	0.9999920001152	17729

Tabella 3.3: Risultati dei test effettuati sul data set 2d5c. Anche qui l'implementazione Jgalilee risulta essere la più efficiente.

I risultati ottenuti con i data set artificiali (Tabella 3.2 e Tabella 3.3) evidenziano che, le tre implementazioni considerate, scoprano esattamente la configurazione di cluster teorica (indice di Rand $\cong 1$). Tuttavia i tempi di computazione indicano che le implementazioni Jgalilee e Yhfyhf sono anche le più efficienti.

Il test effettuato con il data set Indian Pines ci pone ad un bivio: se l'implementazione Jgalilee risulta più accurata in termini di Silhouette, questa

cede il passo all'implementazione Yhfyhf in termini di tempo di esecuzione.
(Tabella 3.1)

In definitiva l'implementazione Jgalilee è stata scelta, in quanto abbiamo prediletto una migliore silhouette ad un miglior tempo di esecuzione.

Capitolo IV

Change Detection

In questo capitolo si descrive l'implementazione dell'algoritmo di change detection facendo uso delle tecnologie discusse in precedenza.

4.1 Introduzione

È stato implementato un algoritmo di change detection pixel-based, di tipo non informativo (non è presente un'informazione di tipo “da a”), sfruttando l’implementazione K-means su MapReduce scelta.

L’idea alla base dell’algoritmo è descritta nel diagramma che segue: (Figura 4.1)

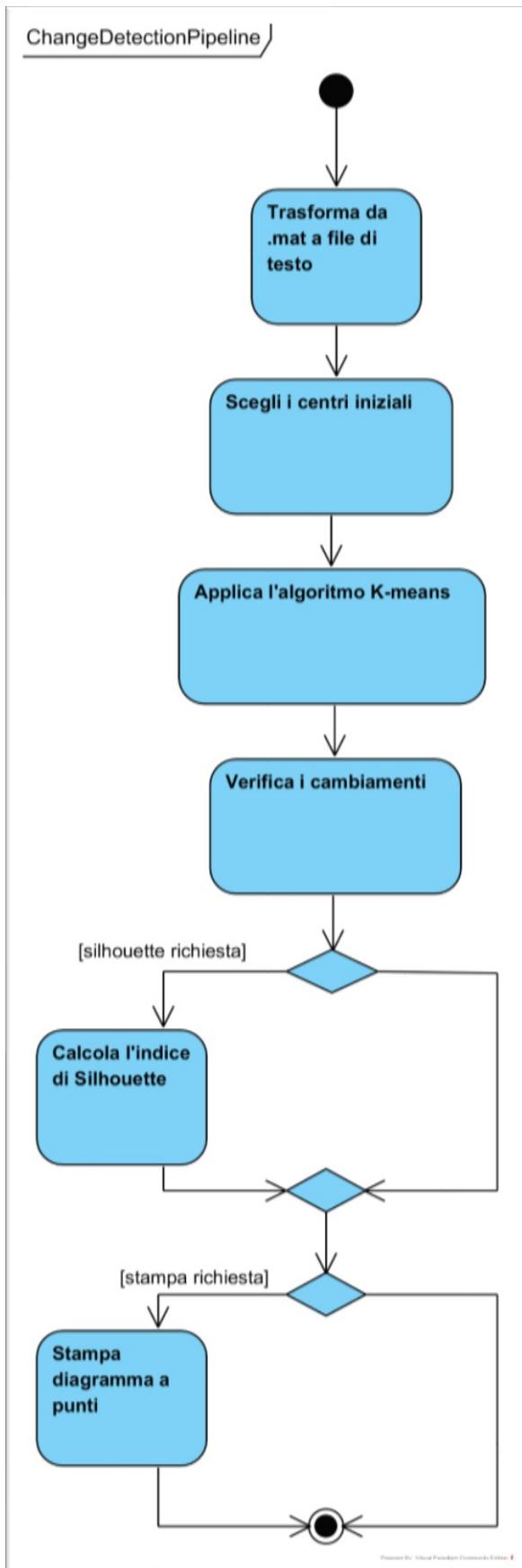


Figura 4.11: Diagramma di attività cloud-level di ChangeDetectionPipeline

Più nel dettaglio, l'algoritmo consiste di 6 passaggi fondamentali, assumendo che i due data set iperspettrali di input siano salvati su mat-file:

1. Trasformare i file .mat in formato testuale compatibile con l'implementazione K-means scelta: otteniamo i data set al tempo t0 e al tempo t1 e la concatenazione dei due;
2. Scegliere un numero di centri in modo casuale dal data set al tempo t0, al fine di creare un bias nel modello, al fine di costruire un modello che “vesta” meglio sul data set al tempo t0;
3. Applicare l'algoritmo K-means sul data set concatenato;
4. Rilevare i cambiamenti confrontando l'associazione pixel-cluster per una qualsiasi coppia di pixel avente stesse coordinate x, y.
5. Calcolare l'indice di Silhouette, se richiesto;
6. Stampare i diagrammi a punti, se richiesto. Un diagramma per il data set al tempo t0 ed uno per il data set al tempo t1.

4.2 Analisi e Progettazione

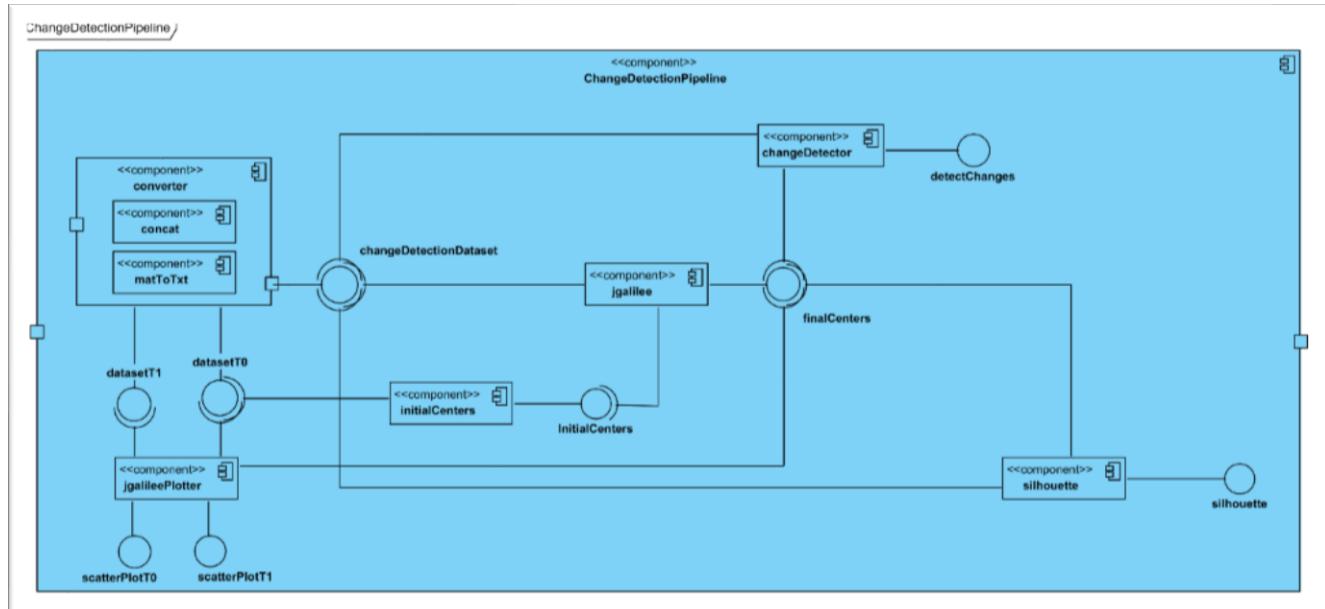


Figura 4.12: Diagramma delle componenti del programma ChangeDetectorPipeline

Il programma ChangeDetectorPipeline è stato costruito facendo interagire varie componenti software: (Figura 4.8)

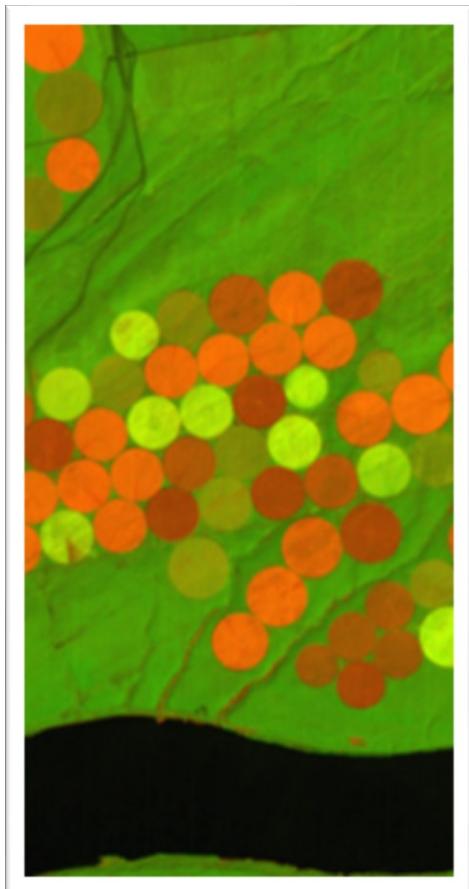
- *converter*: è a sua volta composto dalla componente matToTxt, che si occupa di convertire un elemento del data set bitemporale (ci riferiamo a questi con le espressioni: data set al tempo t0 e data set al tempo t1) in formato di testo, e dalla componente concat, che si occupa di generare un data set in formato testuale frutto della concatenazione di due data set;
- *initialCenters*: sceglie dei punti casualmente dal data set testuale al tempo t0 e li rende disponibili alle altre componenti;
- *jgalilee*: è l'implementazione di K-means su MapReduce scelta. L'operazione di clustering viene effettuata sul data set concatenato e rende disponibile alle altre componenti i centroidi calcolati all'ultima iterazione (o centroidi finali);
- *changeDetector*: considerato il data set concatenato suddiviso in cluster, mediante i centroidi finali e il data set stesso, verifica la presenza di cambiamenti in ogni singolo pixel e li rende disponibili;
- *silhouette*: rende disponibile alle altre componenti l'indice di Silhouette calcolato sul data set concatenato, considerata la configurazione di cluster definita dai centroidi finali;
- *jgalileePlotter*: rende disponibile un diagramma a punti raffigurante un data set tenendo conto della configurazione di cluster definita dai centri finali.

Il flusso di esecuzione avviene come mostrato in Figura 4.1. ChangeDetectionPipeline prende in input:

- RUN ID: l'identificativo dell'analisi che si sta eseguendo;
- DS0: il nome del percorso del mat-file contenente il data set al tempo t0;
- DS1: il nome del percorso del mat-file contenente il data set al tempo t0;

- NAME: il nome della matrice dati nei mat-file;
- K: il numero di cluster che si vogliono ottenere dall'operazione di clustering;
- SILHOUETTE: y se si vuole calcolare la Silhouette, altrimenti qualsiasi altro carattere non-blank;
- PLOT: y se si vogliono stampare i diagrammi a punti dei due data set rispetto all'operazione di clustering precedentemente effettuata, altrimenti qualsiasi altro carattere non-blank.

4.3 Data set multitemporali



*Figura 4.13: Data set Hermiston 2004
Hermiston*

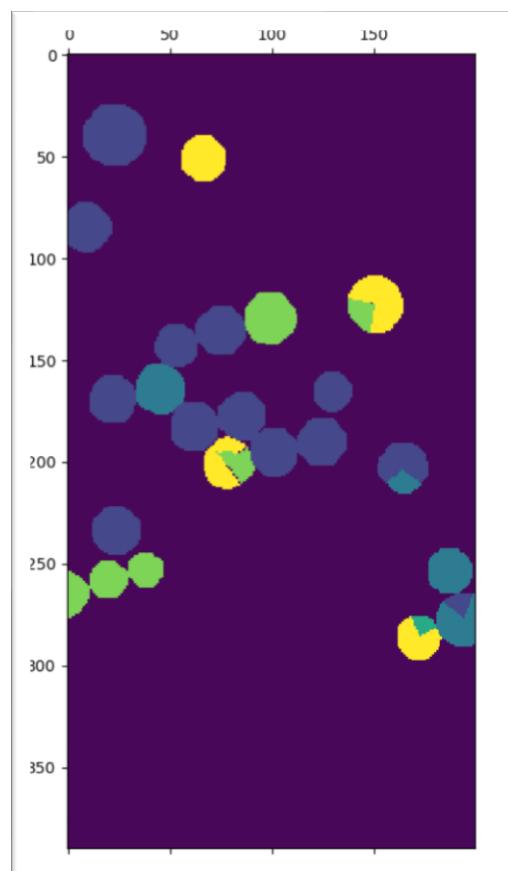


Figura 4.14: Mappa dei cambiamenti del data set multispettrale

Il data set multitemporale Hermiston contiene due immagini, acquisite con il sensore HYPERION, raffiguranti la città di Hermiston (Oregon) negli anni 2004 e 2007. Ogni immagine è composta da 390*200 pixel, ognuno dei quali include 242 bande spettrali. Otteniamo una matrice dati fatta di 78 000 righe e 244 colonne, di queste le prime due definiscono la posizione x, y, le restanti sono attributi spettrali. (Figura 4.2)

La ground truth dei cambiamenti della coppia di data set Hermiston contiene 9 986 cambiamenti riguardanti la coltivazione. (Figura 4.3)

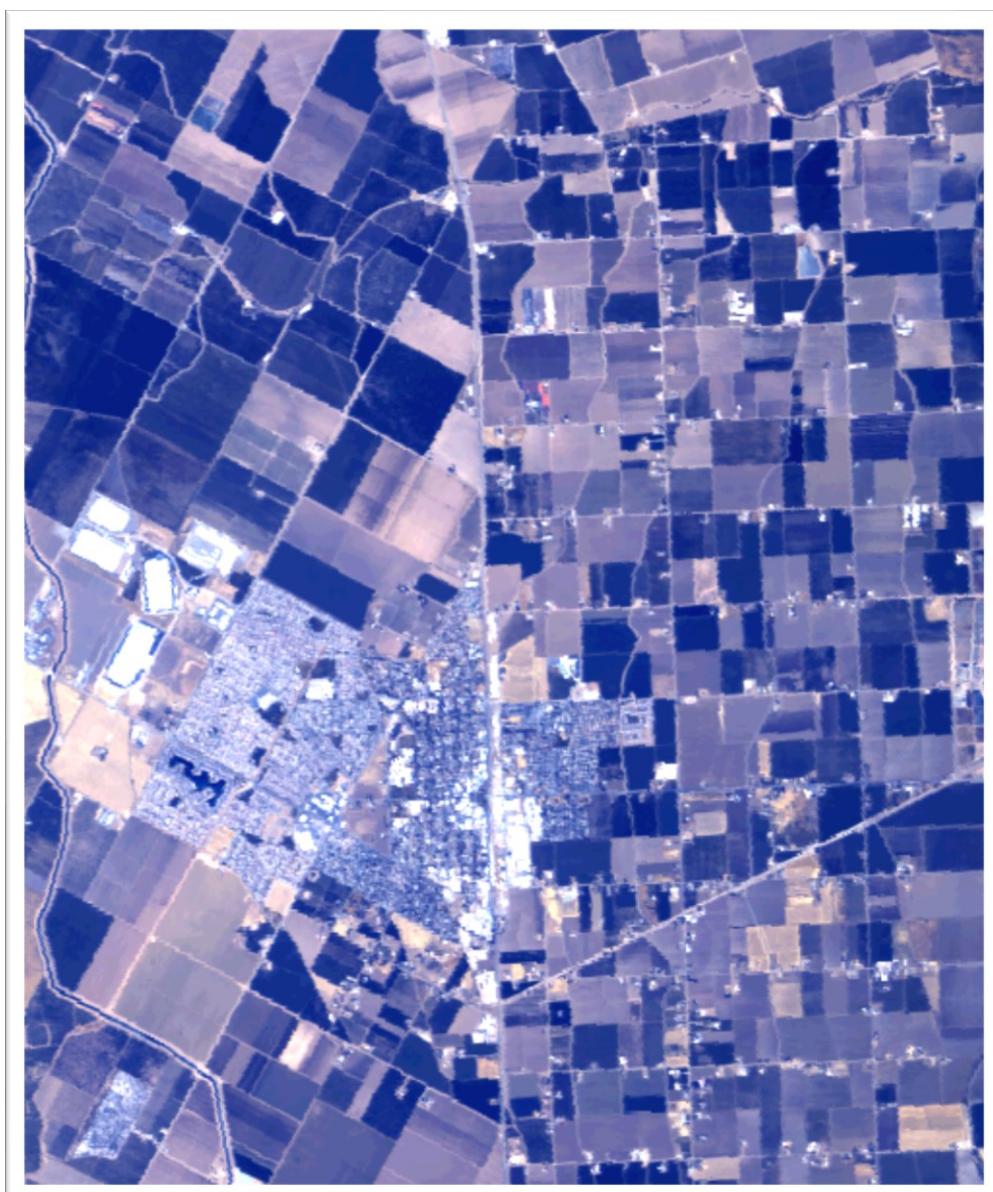


Figura 4.15: Bay area 2013

Il data set multitemporale Bay Area contiene due immagini, acquisite con il sensore AVIRIS, raffiguranti la città di Patterson, in California, negli anni 2013 e 2015. Ogni immagine è composta da 600*500 pixel, ognuno dei quali include 224 bande spettrali. La matrice dati risultante è composta da 300 000 righe e 226 colonne, le prime due colonne contengono la posizione x, y dell'osservazione nella scena acquisita. (Figura 4.4)

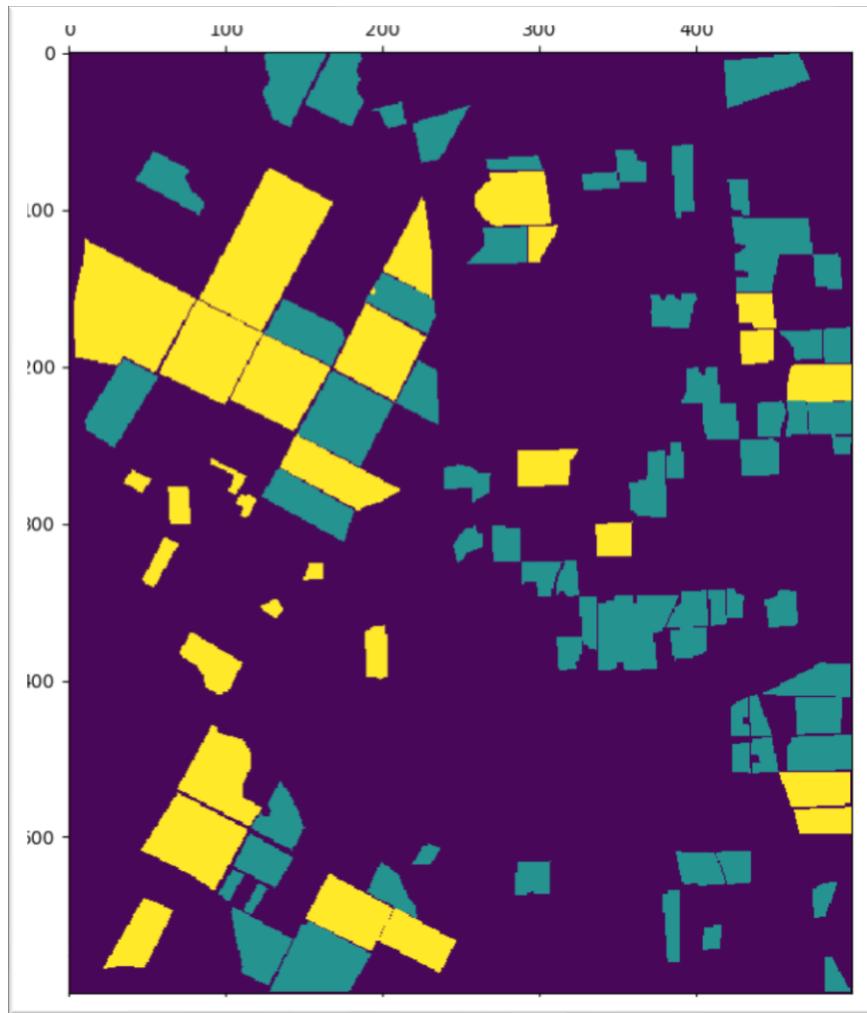


Figura 4.16: Mappa dei cambiamenti della coppia di data set Bay Area

La ground truth contiene 73 481 cambiamenti. (Figura 4.5)

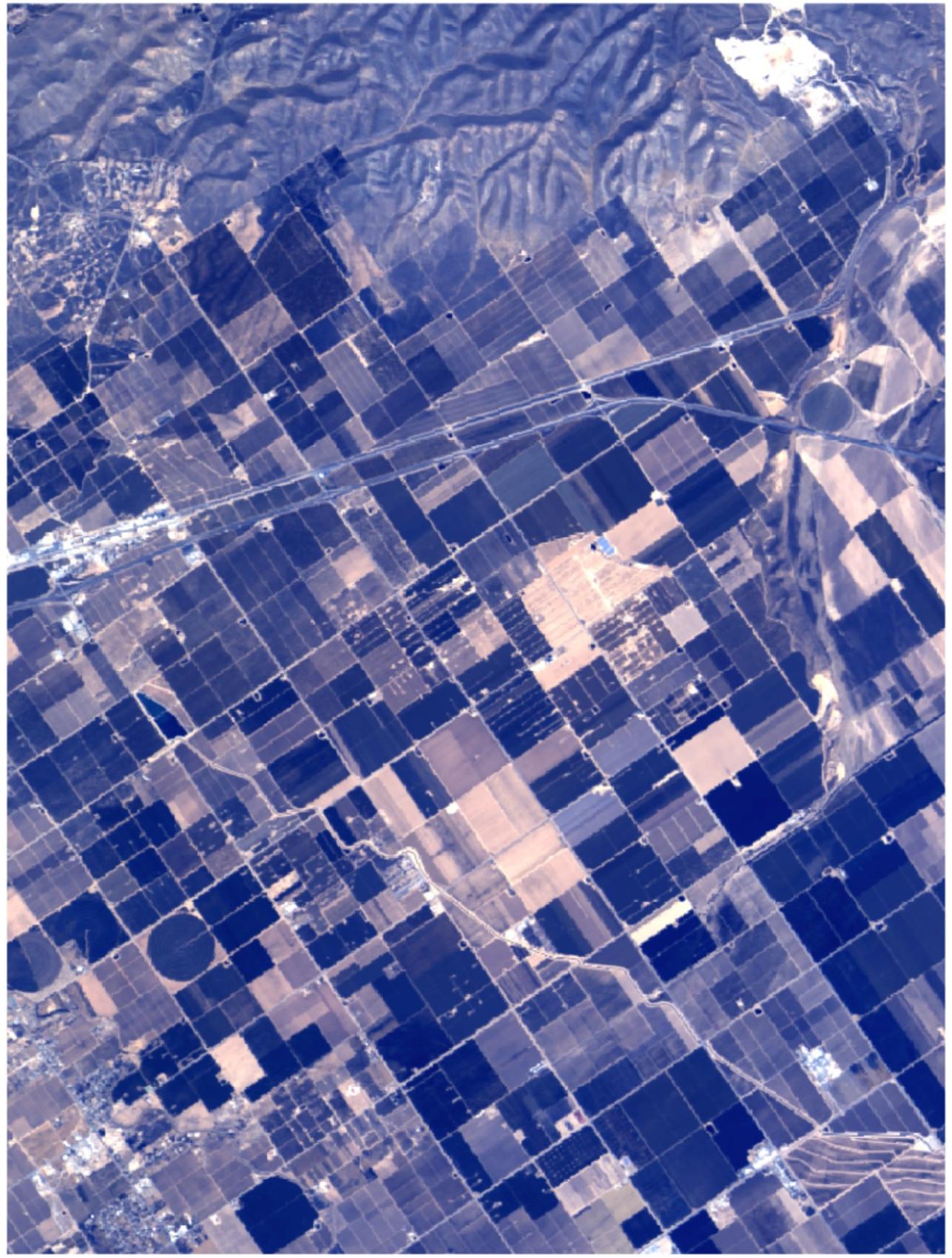


Figura 4.17: Santa Barbara 2013

Il data set multitemporale Santa Barbara contiene due immagini, acquisite con il sensore AVIRIS, raffiguranti la regione di Santa Barbara, California negli anni 2013 e 2014. Ogni immagine è composta da 984*740 pixel, ognuno dei quali include 224 bande spettrali. Segue che la matrice dati è composta da 728 160 righe e 226 colonne, di quest'ultime le prime due contengono, per ogni osservazione, la posizione x, y nella scena. (Figura 4.6)

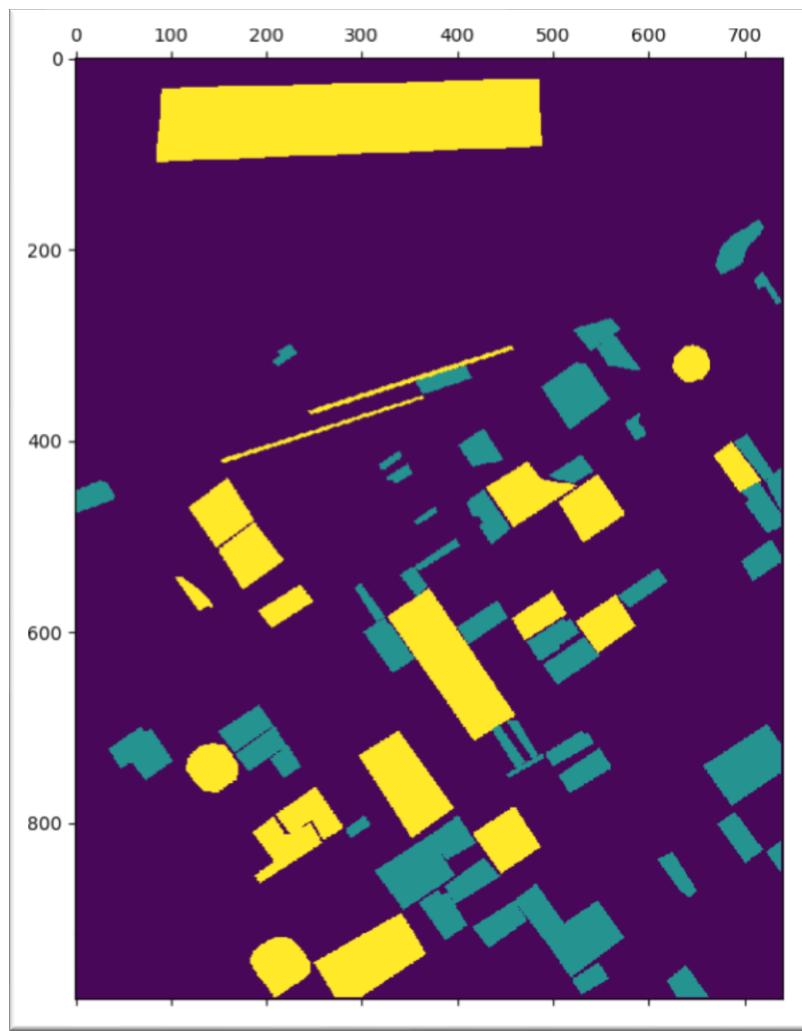


Figura 4.18: Mappa dei cambiamenti del data set multitemporale Santa Barbara

La ground truth contiene 132 552 cambiamenti. (Figura 4.7)

Questi tre data set multitemporali sono disponibili all'indirizzo <https://gitlab.citius.usc.es/hiperespectral/ChangeDetectionDataset/tree/master/>.

Il data set Hermiston è stato utilizzato in un recente studio condotto da Javier

Lopez-Fandiño et al denominato *CUDA Multiclass Change Detection for Remote Sensing Hyperspectral Images using Extended Morphological Profiles* pubblicato durante la 9° conferenza internazionale sull’acquisizione intelligente di dati e sistemi di computazione avanzata: tecnologie ed applicazioni tenutasi tra il 21 ed il 23 settembre a Bucarest, Romania.

4.4 Valutazione

Per valutare l’algoritmo di change detection, sono stati calcolati gli indici di Silhouette per ogni data set considerato. Inoltre sono state calcolate le seguenti metriche:

- *Precision*: definita come $\frac{TP}{TP+FP}$, esprime la proporzione dei cambiamenti rilevati correttamente rispetto a tutti cambiamenti rilevati;
- *Recall*: definita come $\frac{TP}{TP+FN}$, esprime la proporzione dei cambiamenti rilevati rispetto ai cambiamenti effettivamente presenti nel data set multitemporale;
- *Accuracy*: definita come $\frac{TP+TN}{TP+TN+FP+FN}$, esprime l’accuratezza dell’algoritmo considerandola come proporzione dei pixel determinati correttamente (come cambiati o non cambiati) rispetto alla numerosità dei pixel presenti in uno dei data set considerati;
- *F-score*: definita come $2 * \frac{Precision*Recall}{Precision+Recall}$, esprime l’efficacia dell’algoritmo definendo quest’ultima come misura che considera equamente precision e recall.

Dove, con TP (True Positive) intendiamo il numero di cambiamenti rilevati ed effettivamente presenti, con FP (False Positive) intendiamo il numero di cambiamenti rilevati erroneamente, con FN (False Negative) intendiamo il numero di pixel considerati non cambiati ma effettivamente cambiati ed infine

con TN (True Negative) il numero di pixel correttamente considerati non cambiati.

Possiamo meglio interpretare questi valori considerando il seguente test di ipotesi:

$$\begin{cases} H_0: \text{l'osservazione è stata determinata erroneamente} \\ H_1: \text{l'osservazione è stata determinata correttamente} \end{cases}$$

Allora i valori precedentemente descritti possono essere interpretati come segue:

- TP è la potenza del test;
- Ogni osservazione che contribuisce alla determinazione del valore FP è un errore di prima specie;
- Ogni osservazione che contribuisce alla determinazione del valore FN è un errore di seconda specie;

4.5 Risultati

Per ognuno dei tre data set multitemporali considerati, è stato applicato l'algoritmo di change detection utilizzando un numero k=5,6,7 di centri iniziali.

Hermiston	Precision	Recall	Accuracy	F-Score	Silhouette	Tempo
K=5	0.234503	0.628880	0.68966	0.341619	0.55432814	5581695
K=6	0.310240	0.758762	0.753141	0.440408	0.64969196	7271520
K=7	0.168674	0.699779	0.520018	0.271827	0.46992966	8222083

Tabella 4.1: Metriche e indice di Silhouette per il data set multitemporale Hermiston. L'algoritmo di change detection è stato applicato scegliendo un numero k=5,6,7 di centri iniziali

L'algoritmo raggiunge la sua massima efficacia quando applicato con 6 centri iniziali. In tal caso il risultato ottenuto mostra buoni valori di Accuracy e di Recall e la qualità dei cluster ottenuti risulta più che sufficiente (indice di Silhouette), tuttavia è insufficiente il valore di Precision. (Tabella 4.1)

Bay Area	Precision	Recall	Accuracy	F-Score	Silhouette	Tempo
K=5	0.31956935	0.6794409	0.56714	0.43468723	0.3657571261	175214200
K=6	0.3011047	0.6825166	0.53421	0.41786194	0.3392407697	247661315
K=7	0.2887189	0.6837414	0.5099533	0.40599924	0.3136766953	255303884

Tabella 4.2: Metriche e indice di Silhouette per il data set multitemporale Bay Area. L'algoritmo di change detection è stato applicato scegliendo un numero k=5,6,7 di centri iniziali

Per il data set Bay Area l'algoritmo raggiunge la sua massima efficacia quando applicato con 5 centri iniziali (F-score). Il risultato ottenuto, in questo caso, mostra un valore di Accuracy poco al di sotto della sufficienza e un valore più che sufficiente di Recall. La qualità dei cluster (Silhouette) ottenuti per k = 5 risulta tuttavia scarsa, analogamente al valore di Precision. (Tabella 4.2)

Santa Barbara	Precision	Recall	Accuracy	F-Score	Silhouette	Tempo
K=5	0.218401	0.705541	0.486765	0.33355	0.393457076	424635598
K=6	0.21871343	0.7712143	0.45685425	0.3407823	0.368390607	598830093
K=7	0.203467	0.76337	0.412914	0.3212969	0.366183232	458072183

Tabella 4.3: Metriche e indice di Silhouette per il data set multitemporale Santa Barbara. L'algoritmo di change detection è stato applicato scegliendo un numero k=5,6,7 di centri iniziali

Per il data set Santa Barbara, invece, non abbiamo una configurazione di cluster ottimale. Infatti per k = 5 abbiamo un massimo rispetto al valore di Accuracy ma

il massimo in termini di F-Score lo otteniamo per $k = 6$, dove la differenza è dovuta solamente al (significativamente) differente valore di Recall. Tuttavia in entrambi i casi, considerando l'indice di Silhouette e le metriche come un tutt'uno, l'algoritmo non raggiunge la sufficienza. (Tabella 4.3)

Capitolo V

Conclusioni

5.1 Conclusioni

Il remote sensing (o telerilevamento) è diventato, nel corso degli ultimi anni, sempre più importante grazie alle diversificate applicazioni e gli open data forniti da enti europei e non. Per mezzo di algoritmi di change detection è, ad esempio, possibile sfruttare questi dati per lo studio dei cambiamenti climatici o per il monitoraggio del suolo.

Durante la fase di analisi esplorativa sono stati studiati i metodi di change detection, nell'ambito del remote sensing, attualmente in uso. Questi possono essere suddivisi in pixel-based ed object-based; un ulteriore classificazione indipendente considera i metodi “informati”, che prevedono un’informazione di tipo “da a”, e i metodi “non informati”. È stato possibile apprezzare un’altra differenza tra i vari approcci: alcuni di essi sono di matrice statistica, ad esempio l’image differencing, mentre altri sfruttano le conoscenze derivanti dalla geometria, ne è un esempio la change vector analysis.

Il lavoro di tesi è proseguito con l'intento di sviluppare un metodo di change detection che prevede la suddivisione di una immagine telerilevata in un istante di tempo t_0 , in gruppi. Successivamente fare altrettanto con un'immagine raffigurante la stessa scena in un istante di tempo successivo t_1 e stabilire la presenza di un cambiamento, in ogni pixel, verificando che il gruppo di appartenenza del pixel considerato sia cambiato.

A tal proposito sono stati studiati il problema e l'algoritmo K-means. Al fine di rendere l'applicazione di quest'ultimo efficiente, ne è stata considerata la sua variante MapReduce.

La soluzione algoritmica che è stata proposta è la seguente:

1. Concatenare due data set rilevati in istanti di tempo differenti;
2. Applicare l'algoritmo K-means su MapReduce sul data set concatenato;
3. Verificare la presenza di un cambiamento, in ogni pixel, verificando se il cluster di appartenenza varia nei due istanti di tempo considerati.

In seguito sono state esplorate le implementazioni K-means su Hadoop MapReduce presenti nella rete. Di queste è stata decretata la migliore utilizzando indici di Silhouette e di Rand e tempo di computazione come variabili discriminanti. Per fare questo sono stati costruiti due data set artificiali contenti cluster “non sovrapposti” ed è stato inoltre utilizzato il noto data set Indian Pines.

Utilizzando l'implementazione scelta è stato implementato l'algoritmo di change detection proposto in precedenza. Questo è stato valutato facendo uso di tre data set multitemporali raffiguranti le città di Hermiston e Patterson e la regione di Santa Barbara, negli Stati Uniti. Sono state usate metriche come Precision, Recall e Accuracy oltre all'indice di Silhouette e il tempo di esecuzione, al fine di valutare il lavoro svolto.

Gli sviluppi futuri previsti riguardano il confronto dei risultati ottenuti utilizzando differenti distanze e differenti algoritmi di clustering, ad esempio di tipo gerarchico o basato su densità.

Le principali difficoltà affrontate riguardano i problemi di efficienza posti dall'ingente quantitativo di informazioni sottoposte ad elaborazione.

A tal proposito, come prospettiva futura, si intendono acquisire i fondamenti della GPU programming, necessari per migliorare le prestazioni nell'elaborazione di dati vettoriali.

Appendice A

ChangeDetectionPipeline

A.1 Requisiti di sistema ChangeDetectionPipeline

Minimi

Sistema operativo: Linux o Windows 10 con Bash Shell

CPU: Quad-core

Memoria RAM: 6GB

Disco rigido: 5GB di spazio libero

Altri software richiesti: Java Runtime Environment 8 o successiva, interprete Python 3.6 o successivo, librerie Python numpy, scipy e matplotlib.

Consigliati

Sistema operativo: Linux o Windows 10 con Bash Shell

CPU: Octa-core

Memoria RAM: 8GB

Disco rigido: 10GB di spazio libero

Altri software richiesti: Java Runtime Environment 8 o successiva, interprete Python 3.6 o successivo, librerie Python numpy, scipy e matplotlib.

A.2 Primo avvio

È sufficiente estrarre l'archivio ChangeDetectionPipeline in una qualsiasi directory e, da terminale, posizionarsi nella cartella ChangeDetectionPipeline e avviare lo script con la sintassi ./ChangeDetectionPipeline/start.sh.

L'interfaccia utente fornita richiede i seguenti parametri:

1. RUN_ID: l'identificativo dell'analisi che si vuole condurre;
2. DS0: il percorso del mat-file contente la scene all'istante di tempo t0;
3. DS1: il percorso del mat-file contente la scene all'istante di tempo t1;
4. NAME: il nome della matrice nei mat-file contenente i dati;
5. K: numero di gruppi che si vogliono ottenere dall'operazione di clustering;
6. S: y se si vuole calcolare l'indice di Silhouette, altrimenti qualsiasi altro carattere non-blank;
7. PLOT: y se si vogliono stampare su file i diagrammi a punti dei due data set raggruppati, altrimenti qualsiasi altro carattere non-blank;

Al termine dell'esecuzione i risultati dell'analisi saranno presenti nella cartella ChangeDetectionPipeline/RUN_ID.

Bibliografia

[1] SINGH 1989

Singh, Ashbindu (1989) ‘Review Article Digital change detection techniques using remotely sensed data’, International Journal of Remote Sensing, 10: 6, 989 – 1003

[2] MINUS 2015

Minu S., Amba Shetty (2015) ‘A Comparative Study of Image Change Detection Algorithms in MATLAB’, Aquatic Procedia 4 (2015), 1366 – 1373

[3] ERTURK 2015

Alp Erturk, Antonio Plaza (2015) ‘Informative Change Detection by Unmixing for Hyperspectral Images’, IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, VOL. 12, NO. 6, JUNE 2015

[4] BIOUCAS-DIAS 2008

Josè M. Bioucas-Dias, Josè M. P. Nascimento (2008) ‘Hyperspectral Subspace Identification’, IEEE TRANSACTION ON GEOSCIENCE AND REMOTE SENSING, VOL. 46, NO. 8, AUGUST 2008

[5] CHEN 2012

Gang Chen, Geoffrey J. Hay, Luis M. T. Carvalho & Michael A. Wulder (2012) ‘Object-based change detection’, International Journal of Remote Sensing, 33:14, 4434 – 4457

[6] RANDALL 2012

Randal B. Smith (2012), ‘Introduction to hyperspectral imaging’, MicroImages. Inc, 1999-2012

[7] DEVI 2015

R. Naveena Devi, Dr. G. Wiselin Jiji (2015) ‘Change detection techniques – A survey’, International Journal on Computational Science & Applications (IJCSA), VOL. 5, NO. 2, APRIL 2015

[8] HASTIE 2009

Trevor Hastie, Robert Tibshirani, Jerome Friedman (2009) ‘The Elements of Statistical Learning: Data Mining, Inference, and Prediction’ 2nd Edition, February 2009

[9] VENDRAMIN 2010

Lucas Vendramin, Ricard J. G. Campello, Eduardo R. Hruschka (2010) ‘Relative Clustering Validity Criteria: A Comparative Overview’, June 2010, Wiley InterScience

[10] DASGUPTA

Sanjoy Dasgupta ‘CSE 291: Topics in unsupervised learning’ available at <https://cseweb.ucsd.edu/~dasgupta/291-unsup/index.html>.

[11] WHITE 2015

Tom White ‘Hadoop: The Definitive Guide, Storage and Analysis at Internet Scale’, 4th Edition, O’Reilly, April 2015

[12] KARLOFF

Howard Karloff, Siddhart Suri, Sergei Vassilvitskii, ‘A Model of Computation for MapReduce’

[13] ZHAO 2009

Weizhong Zhao, Huifang Ma, Qing He (2009) ‘Parallel K-Means Clustering Based on MapReduce’, CloudCom 2009, LNCS 5931, pp. 674-679, 2009