Hierarchical Cluster Miner

Metodi avanzati di Programmazione – Caso di studio a.a. 2023-2024

Progetto realizzato da: Blanco Lorenzo - Cannito Antonio

Indice

1) Introduzione

Il progetto consiste nella creazione di:

- Un applicativo server, il quale include funzionalità di data mining per la scoperta di un dendrogramma di cluster di dati con algoritmo di clustering agglomerativo, interagendo con la base di dati e comunicando con il client;
- Un applicativo client: costituisce l'interfaccia utente, veicola le richieste dell'utente verso il server

Data una collezione di transazioni, dove ogni transazione è un vettore di valori misurati per una collezione di attributi numerici, ed un intero k, lo scopo del clustering è partizionare la collezione di transizioni in k insiemi di transizioni tali che

- D_i (i=1,…,k) è un segmento (selezione) omogenea di D;
- $D = \bigcup_{i=1}^{k} D_i$ and $D_i \cap D_j = \Phi$.

Assumendo che un esempio è un vettore di numeri reali, la distanza tra due esempi è calcolata tramite la distanza Euclidea. Per partizionare la collezione di transizioni si possono considerare due tipi di distanze tra cluster:

- Distanza single link, la distanza minima tra tutti gli elementi dei due cluster;
- Distanza average link, la distanza media delle distanze tra tutti gli elementi dei due cluster

Per la costruzione del cluster agglomerativo, per ogni livello vengono uniti i due cluster con distanza minima, con distanza una a scelta tra quelle precedenti, fino al livello specificato dall'utente.

Il client una volta caricato il dataset contenuto nel database, può decidere se apprendere il dendrogramma da database oppure caricare il dendrogramma da file.

2) Guida di installazione

Prima installazione: eseguire lo script "setup.bat" presente nella cartella Base/Script/Bat il quale si occupa di creare l'utente, il database, le tabelle e di inserire dei valori di esempio su Mysql, infine esegue l'applicativo server e l'applicativo client (se la password dell'utente "root" per il server Mysql è diversa da "root", modificare il file setup.bat sostituendo la password corretta).

Per le **esecuzioni successive**, eseguire gli script Base/Script/Bat/avvia_server.bat e Base/Script/Bat/avvia_client.bat in questo ordine.

3) Casi di test

***Leggenda**:* Ogni caso di test è composto da delle pre-condizioni, che descrivono i passaggi necessari per raggiungere la sezione del programma da verificare, il flusso di esecuzione che rappresenta il normale funzionamento del codice, le post-condizioni che indicano la parte del programma che si dovrebbe raggiungere in condizioni normali di operatività, infine vi sono gli scenari alternativi che rappresentano alcune possibili situazioni di errore che potrebbero verificarsi e il comportamento previsto del programma in tali circostanze.

***CASO DI TEST 1* :** Creazione della connessione client-server

Pre: E' stata effettuata la prima installazione

Flusso di esecuzione: Eseguire lo script avvia_server.bat, successivamente eseguire lo script avvia_client.bat

Post: Entrambi gli applicativi sono stati avviati correttamente.
Sull'applicativo server si visualizza:

```
Server started : ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8080]

Connessione accettata
```

Sull'applicativo client si visualizza:

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=52695]
Nome tabella:
```

Scenario alternativo : La porta occupata dal server risulta già in uso per un altro processo.

```
Exception in thread "main" java.net.BindException: Address already in use: bind
        at java.base/sun.nio.ch.Net.bind0(Native Method)
        at java.base/sun.nio.ch.Net.bind(Net.java:565)
        at java.base/sun.nio.ch.Net.bind(Net.java:554)
        at java.base/sun.nio.ch.NioSocketImpl.bind(NioSocketImpl.java:636)
        at java.base/java.net.ServerSocket.bind(ServerSocket.java:391)
        at java.base/java.net.ServerSocket.<init>(ServerSocket.java:278)
        at java.base/java.net.ServerSocket.<init>(ServerSocket.java:171)
        at server.MultiServer.avvia_server(MultiServer.java:25)
        at MainTest.main(MainTest.java:15)
```

In tal caso per liberare la porta 8080 da altri processi è possibile usare lo script: Base/Script/bat/free8080.bat.

***CASO DI TEST 2* :** Caricamento del dataset

Pre: Portare a termine con successo CT1

Flusso di esecuzione: il client invia al server il nome della tabella di cui recuperare il dataset ;
il server costruisce il dataset recuperando i dati dal database dalla tabella il cui nome è stato dato dal client.

Post : Sul client si visualizza la scelta del caricamento da effettuare

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=52718]
Nome tabella:
exampletab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:
```

Sul server invece

```
Server started : ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8080]

Connessione accettata
Tabella trovata con successo
```

Scenario alternativo 1) La tabella con nome specificato dal client è inesistente.

Sul client si visualizza

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=52729]
Nome tabella:
exampleerrato
! ! Errore : impossibile trovare la tabella, riprovare
Nome tabella:
```

Sul server si visualizza

```
Connessione accettata
! ! Errore : impossibile trovare la tabella, riprovare
```

Al client viene richiesto di inserire un altro nome di tabella fino a quando il nome è valido.

Scenario alternativo 2) La tabella con nome specificato dal client ha 0 attributi oppure non contiene nessun Esempio.

Sul client si visualizza

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=3031]
Nome tabella:
exampletab5
! ! Errore : la tabella è vuota, riprovare
Nome tabella:
```

Sul server si visualizza

```
Connessione accettata
! ! Errore : la tabella è vuota, riprovare
```

Al client viene richiesto di inserire un altro nome di tabella fino a quando il nome è valido.

CASO DI TEST 3 **:** Il client deve scegliere se caricare il dendrogramma da file o apprendere il dendrogramma da database.

Pre: Portare a termine con successo CT2

Flusso di esecuzione: il client inserisce “1” per caricare il dendrogramma da file o “2” per apprendere il dendrogramma da database

Post:

Se il client sceglie 1, sul client si visualizza

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=52738]
Nome tabella:
exampletab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:1
Inserire il nome dell'archivio (comprensivo di estensione):
```

Se il client sceglie 2, sul client si visualizza

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=5926]
Nome tabella:
exampletab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:2
Introdurre la profondita'  del dendrogramma
```

Scenario alternativo : Il client inserisce una scelta non valida

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=52768]
Nome tabella:
exampletab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:5
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:3
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:
```

In tal caso gli viene richiesto di inserire una nuova scelta fin quando non inserisce una scelta valida

CASO DI TEST 4 **:** Caricamento da file

Pre: Portare a termine con successo CT3 ed aver scelto 1

Flusso di esecuzione: Il client inserisce il nome dell'archivio su cui è memorizzato il dendrogramma da recuperare

Post: il dendrogramma recuperato dal file, viene stampato a video e l'esecuzione del client termina.

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=60570]
Nome tabella:
exampletab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:1
Inserire il nome dell'archivio (comprensivo di estensione):
example3.dat
level0:
cluster0:<[1.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>
cluster4:<[2.0,2.0,0.0]>

level1:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>

level2:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]><[1.0,3.0,4.0]>

level3:
cluster0:<[1.0,2.0,0.0]><[0.0,1.0,-1.0]><[2.0,2.0,0.0]>
cluster1:<[1.0,3.0,5.0]><[1.0,3.0,4.0]>

level4:
cluster0:<[1.0,2.0,0.0]><[0.0,1.0,-1.0]><[1.0,3.0,5.0]><[1.0,3.0,4.0]><[2.0,2.0,0.0]>
```

Sul server si visualizza

```
Connessione accettata
Tabella trovata con successo
Il Client ha scelto di caricare il Dendrogramma da File
Il Dendrogramma è stato caricato con successo
Chiusura connessione Client
```

Scenario alternativo 1) Il client inserisce il nome di un archivio non valido

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=60579]
Nome tabella:
exampletab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:1
Inserire il nome dell'archivio (comprensivo di estensione):
exampleerrato.dat
exampleerrato.dat (Impossibile trovare il file specificato)
Chiusura connessione al Server
```

Sul server si visualizza

```
Connessione accettata
Tabella trovata con successo
Il Client ha scelto di caricare il Dendrogramma da File
exampleerrato.dat (Impossibile trovare il file specificato)
Chiusura connessione Client
```

In tal caso la connessione con quello specifico client si interrompe ma il server rimane disponibile per altre richieste da altri client.

Scenario alternativo 2) Il client inserisce il nome di un archivio in cui è salvato un dendrogramma con profondità maggiore del numero di esempi del dataset caricato.

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=64643]
Nome tabella:
exampletab99
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:1
Inserire il nome dell'archivio (comprensivo di estensione):
prova.dat
! ! Errore : La profondità del dendrogramma salvato nel file scelto (5) è maggiore del numero di esempi nel dataset (2)
Chiusura connessione al Server
```

Sul server si visualizza

```
Connessione accettata
Tabella trovata con successo
Il Client ha scelto di caricare il Dendrogramma da File
Si sono verificati degli errori durante la comunicazione con il client : ! ! Errore : La profondità del dendrogramma salvato nel file scelto (5)
è maggiore del numero di esempi nel dataset (2)
Chiusura connessione Client
```

In tal caso la connessione con quello specifico client si interrompe ma il server rimane disponibile per altre richieste da altri client.

CASO DI TEST 5 **:** Apprendimento da database

Pre: Portare a termine con successo CT3 ed aver scelto 2

Flusso di esecuzione: Il client inserisce la profondità del dendrogramma da costruire ed il tipo di distanza, tra Single Link Distance (1) e AverageLinkDistance (2), che desidera usare.

Post: il dendrogramma viene costruito lato server e viene stampato a video sul client, inoltre viene chiesto al Client il nome dell'archivio sul quale desidera effettuare il salvataggio del dendrogramma costruito.

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=5765]
Nome tabella:
exampletab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:2
Introdurre la profondita'  del dendrogramma
5
Distanza: single-link (1), average-link (2):
1
level0:
cluster0:<[1.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>
cluster4:<[2.0,2.0,0.0]>

level1:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>

level2:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]><[1.0,3.0,4.0]>
```

```
level3:
cluster0:<[1.0,2.0,0.0]><[0.0,1.0,-1.0]><[2.0,2.0,0.0]>
cluster1:<[1.0,3.0,5.0]><[1.0,3.0,4.0]>

level4:
cluster0:<[1.0,2.0,0.0]><[0.0,1.0,-1.0]><[1.0,3.0,5.0]><[1.0,3.0,4.0]><[2.0,2.0,0.0]>


Inserire il nome dell'archivio (comprensivo di estensione):
```

Sul server si visualizza

```
Connessione accettata
Tabella trovata con successo
Il Client ha scelto di apprendere il Dendrogramma da Database
Il Client ha inserito la profondità correttamente
Il Dendrogramma è stato appreso con successo
```

Scenario alternativo 1) Il client inserisce una profondità maggiore del numero di esempi del dataset

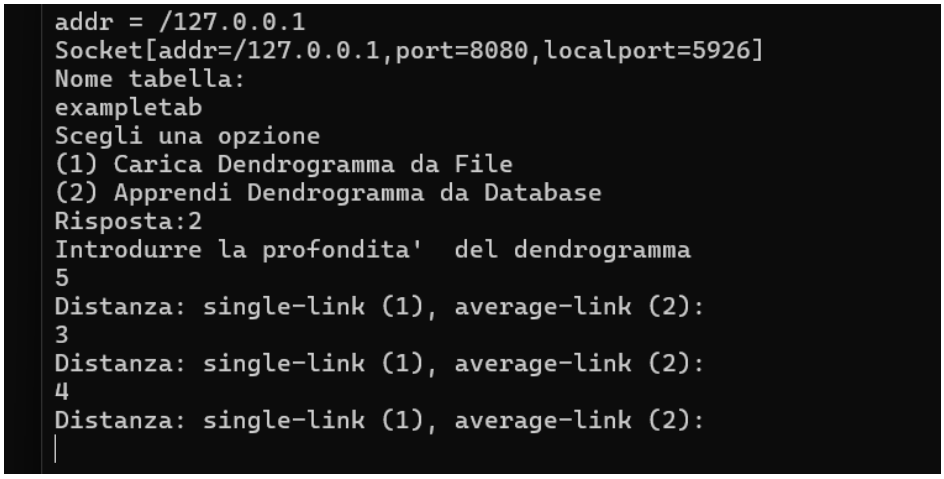
```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=5841]
Nome tabella:
exampletab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:2
Introdurre la profondita'  del dendrogramma
10
Distanza: single-link (1), average-link (2):
1
! ! Errore : la profondità non può essere maggiore del numero di esempi nel dataset (5)
Chiusura connessione al Server
```

Sul server si visualizza

```
Connessione accettata
Tabella trovata con successo
Il Client ha scelto di apprendere il Dendrogramma da Database
Si sono verificati degli errori durante la comunicazione con il client : ! ! Errore :
la profondità non può essere maggiore del numero di esempi nel dataset (5)
Chiusura connessione Client
```

In tal caso la connessione con quello specifico client si interrompe ma il server rimane disponibile per altre richieste da altri client.

Scenario alternativo 2) Il client inserisce tipo di distanza non valida (né “1” per Single Link Distance, né “2” per Average Link Distance)



In tal caso viene richiesto al Client di scegliere il tipo di distanza fin quando non inserice un tipo di distanza valida.

CASO DI TEST 6 **:** Salvataggio su file

Pre: Portare a termine con successo CT5

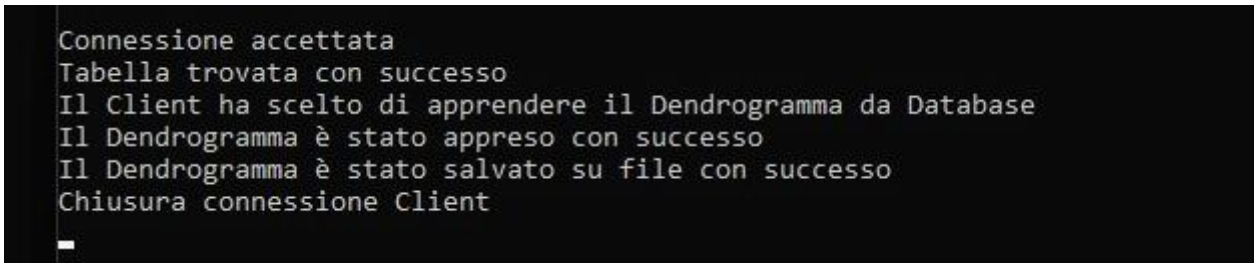
Flusso di esecuzione: Il client inserisce il nome dell'archivio sul quale vuole salvare il dendrogramma.

Post: viene salvato il dendrogramma sul file di nome specificato dal client, la connessione tra client e server viene terminata ma il server rimane disponibile per altre richieste da altri client.

Sul client si visualizza

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=60619]
Nome tabella:
exampletab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:2
Introdurre la profondit  del dendrogramma
5
Distanza: single-link (1), average-link (2):
3
Distanza: single-link (1), average-link (2):
4
Distanza: single-link (1), average-link (2):
1
level0:
cluster0:<[1.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>
cluster4:<[2.0,2.0,0.0]>

level1:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>

level2:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]><[1.0,3.0,4.0]>

level3:
cluster0:<[1.0,2.0,0.0]><[0.0,1.0,-1.0]><[2.0,2.0,0.0]>
cluster1:<[1.0,3.0,5.0]><[1.0,3.0,4.0]>

level4:
cluster0:<[1.0,2.0,0.0]><[0.0,1.0,-1.0]><[1.0,3.0,5.0]><[1.0,3.0,4.0]><[2.0,2.0,0.0]>


Inserire il nome dell'archivio (comprensivo di estensione):
example3.dat
```

Sul server si visualizza



4) Documentazione

Nella cartella Base/Documentazione/javadoc è presente la documentazione javadoc.

Nella cartella Base/Documentazione/uml sono presenti i diagrammi delle classi.