Quality Threshold Clustering

Loconte Lorenzo, M. 683124

June 25, 2019

1 Introduzione

In statistica, il **clustering** è un insieme di tecniche volte alla selezione e raggruppamento di elementi omogenei in un insieme di dati. Le tecniche di clustering si basano su misure relative alla somiglianza degli elementi. Gli algoritmi di clustering raggruppano gli elementi sulla base della loro distanza reciproca.

2 QT-Clustering

QT-Clustering è composto da due applicazioni, un client ed un server. Lo scopo del progetto è permettere a molteplici client di eseguire un algoritmo di clustering su un server utlizzando come sorgente dei dati un database MySQL.

2.1 Server

Il server si occupa principalmente di accettare le richieste dei client. Esso esegue l'algoritmo di clustering vero e proprio utilizzando come sorgente dei dati una tabella (specificata dal client) di un database MySQL. Le funzionalità del server sono le seguenti:

- Inizializzare il server su una porta arbitraria.
- Capacità di gestire le richieste di molteplici client in contemporanea.
- Leggere il contenuto di una tabella di una base di dati indipendentemente dal numero di attributi e dal numero di tuple.
- Eseguire l'algoritmo di **Quality Threshold clustering** usando un raggio arbitrario su un numero indefinito di tuple.
- Fare operazioni di **Principal Component Analysis** per estrarre dalle tuple le componenti principali.
- Salvare su file l'esito della computazione come cache.
- Restituire ai client l'esito della computazione.

2.2 Client

Il client è fornito di una interfaccia grafica che permette ad un'utente di connettersi al server e selezionare la fonte dei dati su cui eseguire l'algoritmo di clustering. Inoltre il client si occupa di rappresentare in forma grafica i cluster o i centroidi all'interno di un grafico bidimensionale.

2.3 Estensione

L'estensione principale rispetto al progetto originale è la grafica per l' applicazione client. Un'altra estensione, sempre a supporto della grafica, è costituita da un'algoritmo di **Principal Component Analysis**.

2.3.1 Grafica

La grafica per l'applicazione client è stata scritta utilizzando le **JavaFX**. L'interfaccia grafica permette all'utente di connettersi al server ed eseguire le operazioni di mining. Inoltre essa mostra i risultati sia in forma testuale che in forma grafica attraverso uno scatter plot.

2.3.2 Principal Component Analysis

La **Principal Component Analysis** (da adesso abbreviato con **PCA**) è una tecnica appartenente alla statistica multivariata usata per estrarre da un insieme di dati N dimensionali M componenti principali in modo tale da ridurre il numero di dimensioni dello spazio in cui si trovano i dati. L'esigenza di fare ciò nasce dal fatto che è molto difficile rappresentare tuple in uno spazio a più di tre dimensioni. Pertanto, tramite algoritmi di **PCA**, è possibile proiettare i dati in uno spazio tridimensionale (o anche bidimensionale) che sia trattabile e soprattuto rappresentabile. Ovviamente la scelta degli assi di proiezione non è arbitraria, infatti punti molto distanti tra loro in uno spazio N dimensionale possono ritrovarsi ad essere molto vicini in uno spazio M dimensionale (con M <N). Generalmente si scelgono gli assi di proiezione in modo tale che il campione proiettato abbia varianza massima. L'algoritmo di **PCA** implementato è il seguente:

1. Si trasforma il campione misto (numerico e discreto) in un campione numerico.
2. Si standardizza il campione.
3. Si calcola la matrice di covarianza del campione.
4. Si calcolano autovalori ed autovettori della matrice di covarianza.
5. Si scelgono come assi di proiezione gli autovettori associati agli autovalori più grandi.
6. Si proietta il campione standardizzato sugli assi di proiezione.

Si tenga presente che questa procedura sacrifica parte dell'informazione del campione originario. Il calcolo delle componenti principali è effettuato dal server al termine dell'esecuzione dell'algoritmo di clustering. E' compito del client poi disegnare sullo scatter-plot i punti. Come libreria per l'algebra lineare è stata usata la libreria **la4j**.

3 Manuale utente

3.1 Client

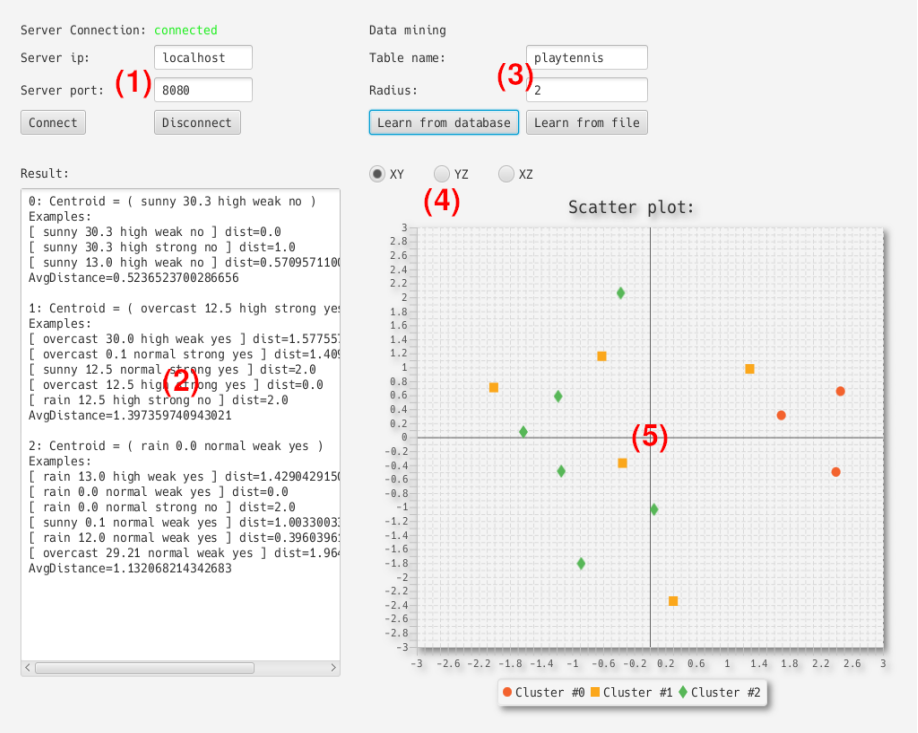

Figure 1: Il form della finestra del client

1. L'utente sceglie IP e porta del server a cui desidera connettersi. Un'etichetta posta in cima al form comunica lo stato della connessione. Se IP o porta risultano non validi si aprirà una finestra di errore. L'utente può scegliere, attraverso i pulsanti, di connettersi o disconnettersi. Quando si è disconnessi e si prova a fare un'operazione verrà mostrato a video un errore.

2. Il risultato, in forma testuale, dell'operazione di data mining. Vengono mostrati i cluster e/o i centroidi e, per ogni cluster, viene mostrata anche la distanza media delle tuple dal centroide.

3. L'utente sceglie nome della tabella del database e raggio da utilizzare. Se la tabella risulta non presente nel database oppure il raggio risulta non valido verrà mostrato a video un errore. L'utente può scegliere se eseguire l'algoritmo di mining partendo da una tabella del database oppure leggere da file. Se il raggio risulta talmente grande tale che il risultato della computazione è un unico cluster oppure il file non è presente sul server allora verrà mostrato a video un errore.

4. L'utente sceglie uno dei tre piani di proiezione dei dati: XY, YZ oppure XZ. In base alla scelta cambierà lo stato dello scatter plot. Si tenga presente che si perde parte dell'informazione delle tuple originarie.

5. Lo scatter plot su cui vengono rappresentati i dati. Ogni cluster (o centroide) possiede un colore ed una etichetta diversa. Al di sotto della griglia è presente una legenda.

3.2 Server

Di default, la porta utilizzata dal server è la **8080**. Tuttavia, al momento dell'esecuzione è possibile specificare la porta da utilizzare da riga di comando come parametro di input al file *Bash/Bat*. Verrà stampato un errore se il formato della porta non è valido oppure se la porta è occupata.

4 Note tecniche

Per il progetto è stato utilizzato lo strumento **git**. Inoltre il progetto è stato hostato su un repository remoto privato su **GitHub**. Come strumento di build system e gestione delle dipendenze è stato utilizzato **Gradle**. In questo modo si ha la completa indipendenza dall'IDE che si intende utilizzare. Durante lo sviluppo sono stati utilizzati i seguenti strumenti:

- *JUnit 4.12*, per i casi di test.
- *JaCoCo*, per la copertura dei casi di test.
- *Checkstyle*, per verificare lo stile.

Sono stati scritti casi di test esclusivamente per il *server* dato che il client è costituito principalmente da codice per la UI. Attualmente la copertura dei casi di test è pari al **74%**. I vari reports (*Javadoc*, *JUnit*, *JaCoCo* e *Checkstyle*) si trovano all'interno di */qt-clustering/reports*. I diagrammi delle classi e dei packages si trovano all'interno di */qt-clustering/documentation*.

4.1 Guida per l'installazione

I file per la distribuzione (file *Bash/Bat* e *.jar*) sia per il client che per il server si trovano in */qt-clustering/distributions*. Per eseguire il server è necessario aver installato MySQL. Dopodichè bisogna accertarsi che il servizo di MySQL

sia in esecuzione. Bisogna poi inizializzare il database MySQL eseguendo lo script *mapdb.sql* presente nella cartella */qt-clustering/distributions/server*. Lo script SQL contiene le istruzioni per creare un database *MapDB* contenente una tabella di esempio chiamata **playtennis** ed un'altra tabella chiamata **test** richiesta per l'esecuzione dei casi di test. Inoltre esso crea un nuovo utente (se non presente) chiamato *MapUser* con diritti di accesso per il database *MapDB*.

4.2 Importare il progetto con Eclipse

Attraverso il tool *Gradle* è possibile, da riga di comando, compilare ed eseguire sia il client sia il server. Inoltre è anche possibile eseguire i casi di test e generare la documentazione *Javadoc*. In alternativa è possibile importare il progetto con Eclipse seguendo i seguenti passi:

1. Importare il progetto come progetto *Gradle*: *File >Import ... >Gradle >Existing Gradle Project*
2. Selezionare la root directory del progetto
3. Cliccare su *Finish*

Verranno così importati due progetti, uno per il *server* e uno per il *client*. Gradle scaricherà automaticamente tutte le dipendenze necessarie. Per eseguire il *client/server* basterà aprire la finestra dei *Gradle Tasks*, selezionare il progetto, e cliccare su *application >run*. Se la finestra *Gradle Tasks* non è presente allora è necessario aprirla selezionando *Window >Show View >Other... >Gradle >Gradle Tasks*. Sempre dalla stessa finestra *Gradle Tasks* è possibile fare altre operazioni, tra cui la generazione del *Javadoc* e l'esecuzione dei casi di test.