Quality Threshold Clustering

Loconte Lorenzo, M. 683124

June 19, 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:

- 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 è composta da quattro sezioni:

1. La sezione per la connessione al server. L'utente inserisce ip e porta del server in ascolto.

2. La sezione che contiene i parametri per il clustering, ovvero la tabella del database da cui ottenere i dati e il raggio utilizzato.

3. La sezione che mostra il risultato della computazione, costituita da una *TextArea*.

4. La sezione che mostra uno scatter plot del risultato della computazione, costituito da uno *ScatterChart* compreso di etichette e simboli diversi per ogni cluster. L'utente può scegliere, tramite pulsanti, su quale piano (XY, YZ, o XZ) proiettare i dati per rappresentarli sullo 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 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. I reports dei casi di test con *JUnit* e di *JaCoCo* sono visualizzabili in */qt-clustering/server/build/reports*. Attualmente la copertura dei casi di test è pari al **74%**. La documentazione *Javadoc* del client si trova in */qt-clustering/client/build/docs/javadoc* mentre per il server si trova in */qt-clustering/server/build/docs/javadoc*. Il diagramma dei packages del server si trova in */qt-clustering/doc/packages.png* mentre il diagramma delle classi in */qt-clustering/doc/classes.png*.

3.1 Importare il progetto con Eclipse

Si può usare *Gradle* direttamente da linea di comando. 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.