

REGRESSION TREE MINER

Caso di Studio del corso Metodi Avanzati di Programmazione (corso A)

Anno 2019-2020

DONATO LERARIO

Matricola: 683240

Indice

1. Introduzione
2. Installazione
3. Casi di test
4. Materiale

1. Introduzione

La regressione si basa sulla predizione di un attributo numerico sulla base dei valori osservati per altri attributi sull'esempio medesimo.

L'apprendimento di una funzione di regressione avviene tramite apprendimento induttivo da esempi forniti. Questi esempi sono strutturati come vettori di coppie attributo-valore e l'attributo di classe (il target) è noto ed è sempre numerico (quando abbiamo a che fare con funzioni di regressione).

Una funzione di regressione è appresa sotto forma di Albero (denominato, appunto, Albero di Regressione), quindi abbiamo i nodi interni che rappresentano le variabili, gli archi fuoriuscenti dai nodi interni che rappresentano un possibile valore per quella proprietà e infine ci sono i nodi foglia che rappresenta il valore predetto per quella classe a seconda dei risultati nelle varie proprietà incontrate nel cammino (path) dal nodo radice (root) al nodo foglia.

Quindi, per costruire un Albero di Regressione abbiamo bisogno di una collezione di insiemi di apprendimento (training Set) dove ciascun esempio è una tupla di valori di un prefissato insieme di attributi (variabili indipendenti) e un attributo di classe numerico (variabile dipendente).

Un attributo può essere continuo (valore numerico) o discreto (valore nominale), ma l'attributo di classe è un attributo continuo e i suoi valori fanno parte dell'insieme dei numeri reali.

I nodi interni dell'albero di regressione sono associati ad un test che coinvolge un dato attributo, e il tipo di test dipende dal tipo di attributo. Poiché se

l'attributo è discreto allora ci saranno tanti archi fuoriuscenti dal nodo interno quanti sono i possibili valori nominali di quell'attributo. Mentre se l'attributo è continuo allora ci saranno solo due archi fuoriuscenti dal nodo interno, quindi il test sarà verificare se il valore dell'attributo è minore/uguale o maggiore di una certa soglia.

Si utilizza l'Errore Quadratico Medio per determinare i migliori attributi che permettono di costruire la funzione di regressione e per determinare il valore della soglia su cui effettuare i test quando abbiamo a che fare con un nodo interno contenente un attributo continuo.

Lo scopo di questo caso di studio è quello di progettare e realizzare un sistema Client-Server denominato "Regression Tree Miner".

In questo programma è presente un Server che istanzia una ServerSocket che rimane in ascolto di connessioni da parte di un Client. Nel momento in cui viene ricevuta la richiesta, la connessione viene stabilita e la comunicazione avviene in un thread dedicato. In questo modo il Server rimane in ascolto ed è in grado di servire più di un Client contemporaneamente.

L'applicativo server permette l'apprendimento di alberi di regressione tramite funzionalità di data mining (estrazione semi automatica di conoscenza nascosta in voluminose basi di dati). Tramite l'apprendimento degli alberi di regressione è possibile ottenere una previsione del risultato.

L'applicativo client è l'interfaccia tra l'utente e il server e, dopo essersi collegato al Server, permette di scegliere la tabella che contiene il training set su cui costruire l'albero di regressione o il file in cui è presente l'albero di regressione precedentemente costruito.

Una volta costruito o caricato l'albero di regressione, è possibile tramite l'applicativo client scegliere la strada da intraprendere all'interno dell'albero (tramite i nodi interni), per poi arrivare al nodo foglia che contiene l'attributo di classe per quel determinato cammino, ossia il valore predetto.

2. Installazione

Per poter avviare il programma, è richiesta l'installazione sulla macchina di MySQL Server e almeno Java SE Development Kit 14, poiché gli applicativi sono stati compilati con questa specifica JDK.

Assicurarsi di avere, oltre all'installazione di una versione recente di MySQL Server, anche il servizio in esecuzione (tramite Gestione delle Attività -> Servizi).

A questo punto, nella cartella Script, troviamo l'SQLScript DropAndCreateDatabase.sql che permette di eliminare versioni precedenti sulla macchina del database MapDB e lo ricrea creando anche un utente MapUser con password "map" che sarà utilizzato per accedere al database. Una volta eseguito questo script SQL, il database viene popolato con le tabelle servo, provaC e prova per poter effettuare dei test con queste tabelle (già piene) nel programma. All'interno della cartella otherSQLscript sono presenti degli script SQL con il nome delle tre tabelle in caso si desideri creare le singole tabelle all'interno del database MapDB senza doverlo eliminare e ricreare.

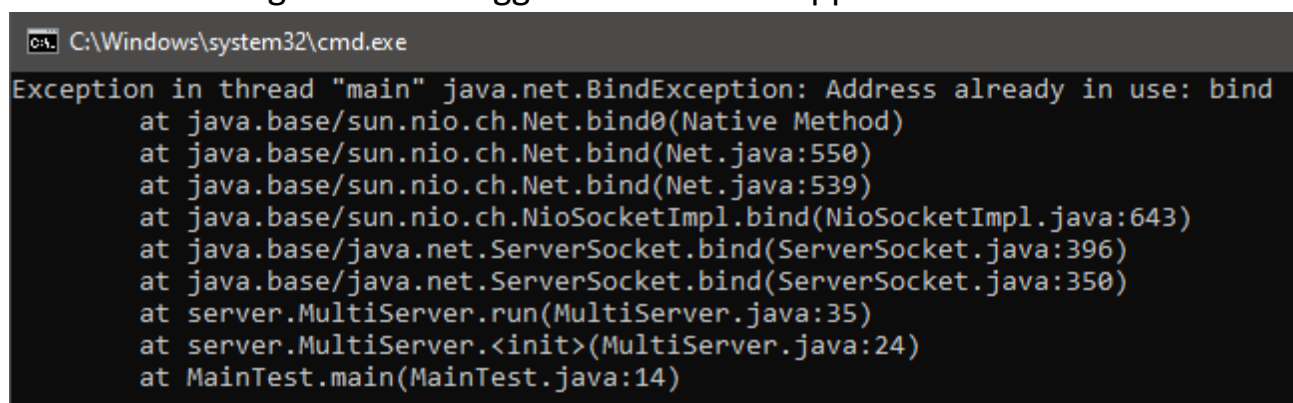
Una volta fatto questo, è possibile cliccare sul file mapServerRun.bat che avvia l'applicativo Server del programma presente nel file jar mapServer. Nel caso in cui la porta 8080 è per qualche ragione già occupata, è possibile avviare il file FreePort8080.bat che libera questa porta e l'applicativo Server non avrà problemi ad avviarsi.

A questo punto è possibile avviare il file mapClientRun.bat che avvia l'applicativo Client del programma presente nel file jar mapClient.

Nel caso in cui ci si voglia collegare ad un Server che non sia localhost, è possibile modificare il file mapClientRun.bat (Tasto Destro -> Modifica) e al posto di "localhost 8080" è possibile inserire l'indirizzo IP e il numero di Porta che desideriamo.

3. Casi di Test

Nel caso in cui la porta 8080 è già utilizzata da qualche processo, viene visualizzato il seguente messaggio di errore nell'applicativo Server:



```
C:\Windows\system32\cmd.exe
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:550)
    at java.base/sun.nio.ch.Net.bind(Net.java:539)
    at java.base/sun.nio.ch.NioSocketImpl.bind(NioSocketImpl.java:643)
    at java.base/java.net.ServerSocket.bind(ServerSocket.java:396)
    at java.base/java.net.ServerSocket.bind(ServerSocket.java:350)
    at server.MultiServer.run(MultiServer.java:35)
    at server.MultiServer.<init>(MultiServer.java:24)
    at MainTest.main(MainTest.java:14)
```

In questo caso avviare il file Free8080.bat nella cartella Script per liberare la porta e riavviare l'applicativo Server.

Se la connessione con il server va a buon fine viene visualizzata la seguente schermata (nell'applicativo Client) dove viene richiesto se apprendere un albero di regressione da un trainingSet presente in una tabella del database MapDB oppure se caricare un albero di regressione salvato su file:

```
C:\Windows\system32\cmd.exe - java -jar mapClient.jar localhost 8080
Socket[addr=localhost/127.0.0.1,port=8080,localport=53297]
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
```

Nel caso in cui si digita un valore diverso da “1” o “2”, la domanda viene ripetuta finchè non viene digitato uno dei due valori (o non si esce dal programma):

```
C:\Windows\system32\cmd.exe - java -jar mapClient.jar localhost 8080
Socket[addr=localhost/127.0.0.1,port=8080,localport=53297]
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
3
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
4
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
```

Digitando “1” viene poi richiesto il nome della tabella dove è presente il trainingSet da cui apprendere l'albero di regressione. Scrivendo, ad esempio, “servo”, viene visualizzata la seguente schermata:

```
C:\Windows\system32\cmd.exe - java -jar mapClient.jar localhost 8080
Socket[addr=localhost/127.0.0.1,port=8080,localport=53318]
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
1
File name:
servo
Starting data acquisition phase!
Starting learning phase!
Starting prediction phase!
0:pgain<=3.0
1:pgain>3.0
_
```

A questo punto verrà creato, nella cartella Script, il file servo.dat contenente l'albero di regressione.

Nel caso in cui scriviamo il nome di una tabella non esistente nel database MapDB, il processo client termina e viene visualizzata la seguente schermata:

```

C:\Windows\system32\cmd.exe
Socket[addr=localhost/127.0.0.1,port=8080,localport=50997]
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
1
File name:
nometabellanonexistenteneldatabase
Starting data acquisition phase!
Errore nell'acquisizione delle Transazioni nella Tabella
Interrotto

```

Continuando con la scelta dei nodi, porteremo a termine il programma. Quindi ora è possibile provare il programma caricando un albero di regressione precedentemente salvato su file. In questo caso, dopo il collegamento tra applicativo Client e applicativo Server, arriviamo alla seguente schermata vista precedentemente:

```

C:\Windows\system32\cmd.exe - java -jar mapClient.jar localhost 8080
Socket[addr=localhost/127.0.0.1,port=8080,localport=53297]
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]

```

A questo punto digitiamo “2” e inseriamo il nome del file presente nella cartella, in questo caso “servo” e clicchiamo. Ci troveremo di fronte a questa schermata:

```

C:\Windows\system32\cmd.exe - java -jar mapClient.jar localhost 8080
Socket[addr=localhost/127.0.0.1,port=8080,localport=53349]
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
2
File name:
servo
Starting prediction phase!
0:pgain<=3.0
1:pgain>3.0
_

```

Scrivendo il nome di un file non presente nella cartella Script, verrà visualizzato il seguente messaggio:

```

C:\Windows\system32\cmd.exe
Socket[addr=localhost/127.0.0.1,port=8080,localport=51019]
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
2
File name:
nomefilenonesistente
nomefilenonesistente.dmp (Impossibile trovare il file specificato)
Interrotto

```

Se, durante la computazione, viene selezionata una scelta non indicata, verrà visualizzato il messaggio:

```
C:\Windows\system32\cmd.exe - java -jar mapClient.jar localhost 8080
Socket[addr=localhost/127.0.0.1,port=8080,localport=51023]
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
2
File name:
servo
Starting prediction phase!
0:pgain<=3.0
1:pgain>3.0

3
The answer should be an integer between 0 and 1!
Interrotto
Would you repeat ? (y/n)
_
```

A questo punto digitando “y” (ossia Yes) il programma ricomincerà dall’inizio della fase di predizione, mentre digitando “n” (ossia No) il processo Client verrà terminato.

Completando con successo la computazione, otterremo la Predicted class, ossia il risultato della funzione di regressione. E infine ci verrà chiesto se ricominciare da capo.

```
C:\Windows\system32\cmd.exe - java -jar mapClient.jar localhost 8080
Socket[addr=localhost/127.0.0.1,port=8080,localport=53355]
Learn Regression Tree from data [1]
Load Regression Tree from archive [2]
2
File name:
servo
Starting prediction phase!
0:pgain<=3.0
1:pgain>3.0

1
0:screw=A
1:screw=B
2:screw=C
3:screw=D
4:screw=E

4
0:vgain<=2.0
1:vgain>2.0

1
Predicted class:0.6343763033333333
Would you repeat ? (y/n)
```

4. Materiale

- Codice Sorgente (cartelle MapClient e MapServer)
- Javadoc
- Script SQL per la creazione del database MapDB, delle tabelle e per il loro riempimento (DropAndCreateDatabase nella cartella Script)
- Singoli script SQL solo per la creazione delle tabelle all'interno di MapDB (cartella Script -> cartella otherSQLscript)
- File Jar contenente gli applicativi Client e Server (mapClient.jar e mapServer.jar, nella cartella Script)
- File bat per avviare gli applicativi Client e Server (mapClientRun.bat e mapServerRun.bat, nella cartella Script)
- Diagramma delle classi UML (cartella UML)