Analisi Database Crime Boston (sito kaggle)

Il database contiene 327.821 record e 17 colonne.

Colonna delle UCR offence definitions: tipologie di crimini che in america vengono divise in categorie. Diviso in 3 gradi di gravità

UCR Part I sono i crimini più pesanti come omicidio, stupro, ecc (<https://www.ucrdatatool.gov/offenses.cfm>) (<https://en.wikipedia.org/wiki/Uniform_Crime_Reports#UCR_crime_categories>)

Indicazioni pratiche:

Utilizzo la colonna del **offense\_code\_group** per identificare le tipologie, sono 67 categorie!

Part One: 9 categorie

Con le colonne **lat** e **long** ho la posizione degli eventi e con **occurred\_on\_date** ho la data e ora.

PROBLEMI RISCONTRATI:

nei dati latitudine e longitudine mancano dei record, in particolare o non c’è proprio il dato o segnala -100.000.000. TOLTI (nel database ufficiale)

Dati sono discontinui nel tempo nel senso che nel 2015 sono presenti solo i mesi di luglio e agosto. Nel 2016 e 2017 mancano dei mesi. Il database si ferma a ottobre 2018.

Nel 2016 manca: marzo, aprile, maggio, giugno, novembre, dicembre

Nel 2017 manca: febbraio, marzo

Nel 2018 manca: novembre e dicembre perché si interrompe prima

* Vi sono dei record che hanno stessa location, stesso time, diverso tipo di evento e conseguente stesso incident number, (credo che identifichino uno stesso evento ma è stato inserito più volte lo stesso elemento)

Struttura della mia tabulazione:

0.Incident\_Number, 1.Offence\_code\_group, 2.date, 3.UCR\_part, 4.lat, 5.long

Per neighborhood:

ASSUNZIONI IMPORTANTI:

parser di Location e Time basate sulle stringhe particolari in input che ho (i vari caratteri della stringa), se si cambia zona di riferimento non funzionano!! (dovrei fare un altro tipo di parser)

ho reso intere le coordinate (10 cifre senza virgola) sottratte tra di esse e valutate il raggio, (raggio bene tenerlo sulle 7 cifre di range)

tempo ho trasformato in giorni e sottratto (attraverso la classe datatime)

Creato altra funzione che calcola meglio la distanza (file “calculate\_Location”) però non so se vale la pena utilizzarlo

25/03/19

Per implementare il “Set of instances” utilizzo la libreria set di python, ogni elemento è dato dal “incident number” del mio database inserita o no in base all’algoritmo

Ho impostato come calcolare il set of instance ma non sono riuscito a concluderlo

27/03/19

Ho risolto parte del problema di “Set of instances”, adesso credo funzioni ma da problemi con i doppioni in quanto “Incident number” non è sempre univoco ha delle ambiguità. L’algoritmo ci mette sempre un tempo elevato.

Es. [Larceny, Homicide] ci mette 48 secondi circa

Possibile soluzione: creare un dataset in qualche modo tridimensionale con latitudine, longitudine e tempo come assi. In modo che valuto solo la zona di spazio interessata senza iterare tutti gli altri eventi inutilmente. Non è implementabile come matrice in quanto ci sarebbero tante celle vuote perché soprattutto per il tempo non sono così frequenti questo tipo di eventi

Altra possibile soluzione: salvo i dati in un database e faccio in modo di interrlogarlo in modo efficiente con sql (possibile collaborazione con Maurino)

27/03/19

COLLOQUIO PROF – cose da dire:

* Raccontare come ho calcolato la distanza spaziale e temporale
* Neighborhood e Set of instances
* Spiegare problem del dataset:
  + Grosse dimensioni
  + Selezione dei dati utili
  + Concentrato sul 2018
  + Mancanza di dati
  + Errori dei dati
* Spiegare problema dei tempi con calcolo set instances, chiedere suggerimenti
* Possibile collaborazione con progetto Maurino
* Dire cosa pensi di fare ora

CONSIGLI PROF:

ok maurino – progettone di complementi

capire se mantenere la struttura albero consigliata o utilizzare qualcosa di già presente che mi faciliti il problema! Non stravolgere il database ma magari concentrarmi solo su un sottoinsieme di record significativi

Cerca altre strutture dati possibili (red-black trees)

Cerca algoritmo a priori (struttura ad albero)

Association rule mining, in questo tipo di algoritmi vi è:

Ricerca dei candidati utilizzabili

Vedere se rispettano una certa regola per capire se sono significativi

(nota: distanza spaziale forse tramite maps per le strade ecc)

Cosa ho fatto:

* Preparato la funzione per il calcolo del PR e successivo calcolo PI

COSE PER DOMANI:

cercare strutture dati alternative (vedi sopra)

provare a implementare la struttura dati del paper usando le classi python e collegamenti con liste ecc

MAURINO:

* Implementa il csv in base di dati relazionale con indici per velocizzare la ricerca all’interno della stessa
* Cerca se vi sono indici in due dimensioni per mySQL
* postgres, mongDB come soluzioni per rappresentare i punti nello spazio
* vortex (per rappresentare i dati in 3 dimensioni)
* grafo delle distanze per rendere più efficiente il tutto
* pensare a grafo delle distanze viarie per tenere conto di quanto tempo si impiega da raggiungere un luogo a un altro

31/03/19

Ho creato il database mySQL di 10.000 records con incident number univoco e si chiama ‘crimedata2018’

Gli altri:

* test2018: contene parte dei dati del 2018 ma è inconsistente rispetto la dimensione delle colonne
* newtable: prova di upload di csv
* crimedataset: contiene tutti i dati del 2018 con UCR-type=One ma con i doppioni sul incident number

PER i miei test posso utilizzare il **‘crimedata2018’**

03/04/19

Sono riuscito a far interrogare a python il database e creato il file partRatioSQL.py dove ho riprogettato tutto in modo che interroghi il database mysql piuttosto che il file csv.

La velocità mi sembra migliorata ma non considerevolmente.

PER MIGLIORARE: credo che dovrei rimodellare il database in modo da fargli calcolare la distanza già prima della query così da migliorare le tempistiche (credo) per questa cosa però dovrei strutturare in modo diverso i dati, in particolare la data e la location

Potrei provare con solo 1000 record così riesco a fare test più significativi

! DA FARE: strutturare meglio i file di test e di codice

! DA FARE: fare in modo che il mysql.cursor sia passato come parametro, in questo modo lo rendo più scalabile

Performance:

* SQL [Aggravated Assault, Residential Burglary] – 3 min

INFLUENZA la performance:

Ordine all’interno della sequenza di tipi. La cardinalità del primo tipo di evento. Il numero di record. Il numero delle classi di eventi

Per ora non mi metto più a ottimizzare la base di dati. Prendo meno record ma significativi quindi:

NUOVI DATASET:

* Dataset2018\_2: preso ogni 7 un record da dataset2018, eliminato righe che hanno part = other (11.497)
* Dataset2018\_2\_one: preso i record di tipo one da dataset2018\_2 (2132) – 9 tipi di eventi
* Dataset2018Small: preso ogni 60 record uno da dataset2018 (1150)

NUOVO DATABASE:

crimedata2018small: preso i dati da dataset2018small

aggiunta della chiave primaria, indice su offence\_code\_group, occurred\_on\_date, latitude, longitude

tempi delle queys: accettabili, si rallenta se uso Larceny o aggravated assault, in quanto sono presenti in numero molto maggiore rispetto alle altre categorie

fatto la funzione che calcola il PI, lo fa in tempi ragionevoli per cui mi fermo qui con l’ottimizzazione (per ora)

08/04/19

Sono stato dal prof e mi ha detto:

* Knn: studiare come funziona per capire come ottimizzare l’algoritmo di ricerca del vicinato e set of instances
* Provare a implementare la struttura dati del paper in modo da suddividerla in levels. Per cercare di renderla il più efficiente possibile

Iniziato a implementare SPTree. L’idea è fare una list per ogni layer e mettere queste list in un vettore in modo che so a quale livello sto lavorando e posso salire e scendere dai vari livelli.

DA FARE: vai avanti con SPTree

18/04/19 (e giorni precedenti)

Ho implementato SPTree con una logica diversa rispetto alla precedente, ogni nodo ha:

* Parent1 = punta al nodo parent1
* Parent2 = punta al nodo parent2
* Value = valore del carattere che sto considerando
* Children = lista di puntatori a nodi

NB: questi puntatori (gestiti da python) puntano direttamente agli oggetti nodo originali, per cui se questi vengono modificati la modifica si vede anche attraverso l’albero!

Ho pensato a questa soluzione in quanto mi sembra la più semplice. Ora dell’albero ho bisogno solo del puntatore alla root poi posso navigarlo scendendo attraverso la lista children!!

Caso base (root e prima riga dell’albero):

root: Node(None, None, None, Lista primi figli che sono tutti i tipi di eventi)

tipi: Node(root, root, value, children)

ho dovuto settare come second parent la root per far funzionare ricorsivamente i second parent sotto. In quanto per il second parent la sua assegnazione è fatta partendo dal nodo da cui dovrà nascere il figlio (vedi dopo)

Inserimento:

PREMESSA: io inserisco le sequenze dalle più brevi alle più lunghe quindi parto da quelle lunghe 2 (quelle iniziali le inserisco nell’inizializzazione dell’albero) poi quelle da 3 e via di seguito.

Quindi se ho A🡪B e A🡪C non posso inserire A🡪B🡪C in quanto B🡪C non esiste!!!!

Assumo quindi che della sequenza che devo inserire in sostanza devo aggiungere solo l’ultimo carattere, il resto devo averlo già inserito, quindi:

itero sulla sequenza fino al penultimo carattere

itero sui figli del nodo corrente (partendo dalla root)

aggiungo il nuovo nodo con (parent1: nodo corrente, parent2: searchParent2, value, children:None)

searchParent2:

parto dal nodoCurr.parent2

cerco il nodo che ha value == nuovo value e prendo questo nodo come second parent

funzione search Sequenza:

scendo nell’albero attraverso i children seguendo la sequenza in input. Return il nodo finale di arrivo

24/04/19

Oggi mi sono concentrato sulla parte di algoritmo della Generate Candidates

Ho definito la funzione che la calcola. Ho dovuto aggiungere una nuova funzionalità al SPTree.

Ho aggiunto un attributo chiamato **candidates**. **Candidates** è una lista che contiene tutte le sequenze presenti nell’albero, in questo modo mi è più facile ritrovarle senza dover passare tutto l’albero. Nel caso in cui io elimini una parte dell’albero triggero un metodo refreshCandidates che riaggiorna la lista in modo da eliminare le sequenze che ho tagliato fuori (grazie a searchSeq)

Inoltre ho migliorato un po’ la stampa e aggiunto diversi test! (testPaper, TestDataset su tutti)

Cose da fare!

* Capire come inserire in automatico le sequenze di lunghezza superiore [vedi algo 6]
* Capire se calcolarsi il neighborhood di pari passo come nell’algoritmo del paper (quindi salvarsi il setInstances in ogni nodo oppure boh)!!
* Sistemare meglio il codice in modo che sia correlato al paper

25/04/19

**Idea:** aggiungere a ogni nodo dell’albero il set of instances corrispondente. In questo modo contare è rapido e quando aggiungo un figlio, prendo il set del padre e faccio il neighborhood con il figlio senza dover ricalcolare tutta la sequenza. Ci guadagno in tempo ma ci perdo in spazio per ogni nodo. Non credo crei problemi per python.

Da fare: generatore di tutte le coppie possibili di sequenze

Capire se ha senso la parte sulle sequenze salvate all’interno del tree

Aggiunto il set come parametro del nodo di un albero. Creata insertSet per settare il set di un dato nodo. L’idea è quella di creare il primo livello dell’albero con il set settato a None. Poi tramite la searchNode trovo il nodo dove inserire tramite la insertSet il set corrispondente. (così non devo far collegare al database l’albero

27/04/19

Fix dell’albero.

Ora per creare l’albero il costruttore inizializza soltanto la root e i candidates, e fixato l’insert node per inserire in root. In questo modo gli inserimenti sia dei tipi che dei set per tipi è fatto esternamente

Sistemate tutte le funzioni a esso collegate!

To Do: sistemare PI e PR che viene calcolato in base a come scendo nell’albero

30/04/19

Ho fatto il metodo calculatePR tramite l’albero e fatto il test associato.

In input prende il nodo sul quale devo predere la setIns e il tipo per calcolarlo, nonché dell’albero di partenza!

To Do: fare PI basandosi sulla PR appena creata

02/05/19

Aggiunto metodo computePI per calcolo del PI attraverso l’utilizzo del nuovo albero. Valuto i 2 casi:

old tree: 1.08 min

new tree: 0.49 min

risparmio tempo nelle sequenze più lunghe di 2 infatti l’old tree si doveva ricalcolare tutto il set of instance dell’intera sequenza, invece con quello nuovo il set lo calcola solo sulla parte aggiunta nuova

03/05/19

Fixato il codice adesso funzinona bene la parte di generazione anche di sequenze con 3 tipi. Da creare il controllo interno alle sequenze (non devono contenere lo stesso tipo 2 volte o sì?)

To do: sistemare il codice in modo che sia presentabile e scrivere un po’ una relazione sul suo funzinoamento

06/05/19

Fare metodo che valuta se all’interno di una sequenza candidata (o che si sta valutando) ci siano o meno tipi di eventi ripetuti, e nel caso eliminarla

RISOLTO aggiunto metodo checkDouble che effettua il controllo e applicato al candidateGenTree!!

Iniziato a scrivere la verifyTopCandidates che fa la verify di candidates lunghi lun, con un certo teta, con un vincolo n di elementi da restituire, in modo da avere i più significativi!

Devo sviluppare la parte di deleteBadSeq (vedi foglio)

07/05/19

Completato il metodo verifyTopCandidates che fa la verify delle top n sequenze

Scritto la stbfmTop che utilizza il metodo verifyTopCandidates ma no funziona nel modo esatto in cui dovrebbe, mi inserisce più elementi di quelli che gli ho specificato

Risolto quel problema ma ora mi elimina troppe sequenze che non ritrova

Dovrei aver risolto il problema dividendo l’eliminazione delle sequenze nella lista top e lista candidates

Cose da dire al prof:

nuova struttura ad albero utilizzata:

* Utilizzato una filosofia di puntatori tra oggetti, in questo modo basta modificare un nodo che lo legge modificato anche l’albero
* Nodo con set, parent2 funziona e lo uso nella generate
* Candidates, ha senso (spero) io lo utilizzo per tenere in memoria tutte le sequenze accettate dall’albero

Divisione della computazione tra candidates e verify, spiego perché e aggiunta del threshold

Costruzione della verifyTop e come ho gestito i vari casi, in particolare che io mi baso sempre sui miei candidates invece che su Lm come il paper, per comodità e per buon senso, problema non avendo lm non so quando fermarmi nel generale (capire come fare – basta valutare se ha inserito o meno delle nuove sequenze)

Io passo il top di volta in volta al livello successivo.

Specificare che per ora io non ho implementato l’algoritmo 1 di base per cui dovrebbe funzionare fino al livello vuoto

Performance: con 19 sequenze lunghe 2, teta = 0.25, n = 20, 3 livelli di albero, r = 0.1 km, t=2 giorni

time: 5.23 min, (2.34 per lv1-2)

DA FARE:

Apriori algo decision trees

Fp grow algorithm

Capire la logica di questi algoritmi e studiare come possono migliorare la nostra computazione

E verifico i risultati correttamente

09/05/19

Analisi apriori

If <x> not in <y> costrutto per valutare se è presente o meno un elemento in un altro

Devo capire come gestire il Ck se farlo separato e non tutto insieme come in cadidates, meglio perché gestisco tutto come nel paper!

Push lo stato attuale poi modifico il file

SPTree3 !! in questo nuovo albero tolgo i candidates che dovranno essere gestiti esternamente!!

TODO: devo sostituire i commenti di eliminazione rispettando l’algoritmo del paper!! In particolare stai attendo a quale delle liste elimino

E la return [top, seqlist(lungx)]

11/05/19

Ho sistemato finalmento SPTree3, albero in cui non ho i candidates

Ho sistemato l’intero codice in modo che funzioni come prima e sfrutti le informazioni precedenti (MOLTO PIÙ SIMiLE AL PAPER).

candidateGenTree(candidates, tree)

chiede in input la lista di candidati di lunghezza l-1 (non effettua più questo controllo) genera le nuove sequenze accettate dall’albero in input e le restituisce al chiamante sottoforma di lista

verifyTopCandidates(candidates, teta, top, num, tree)

ha in input i candidate da valutare, il teta da applicare, le top sequenze trovate finora, il numero di elementi da considerare e l’albero della computazione

questo restituisce i futuri candidati da considerare accoppiati con i top(n) trovati finora, ovvero:

return [seqFuture, [seqTop, piTop]]

STBFMinerTop: considera questi nuovi metodi e si comporta come l’Algo[1] del paper

Performance: con 19 sequenze lunghe 2, teta = 0.25, n = **10**, 3 livelli di albero, r = 0.1 km, t=2 giorni

time: 2.25 min, (1.20 per lv1-2)

Performance: con 19 sequenze lunghe 2, teta = 0.25, n = **20**, 3 livelli di albero, r = 0.1 km, t=2 giorni

time: 2.42 min, (1.21 per lv1-2)

Performance: con 19 sequenze lunghe 2, teta = 0.25, n = **30**, **4** livelli di albero, r = 0.1 km, t=2 giorni

time: 3.47 min, (1.21 per lv1-2)

DATABASE:

aggiunto indici a offence\_code, latitude, longitude, data (per ottimizzare il tutto)

nuova tabella:

crimedata2018\_4000: contiene crimini del 2018 (part one) 4270 record (indici come sopra)

TODO:

verificare i risultati (scrivili da parte e verificali uno a uno – foglio excel)

Aggiungere indici alle tabelle crimedata2018small e crimedata2018

Calcolo delle performance delle nuove configurazioni

14/05/19

Finite le verifiche dei dati, vedi foglio excel (verifyData) sistemato codice vecchio sul insSet sbagliato per validare i dati, indici al database già aggiunti

TODO: scrivere cosa ho capito degli algoritmi che mi ha chiesto il prof e iniziare a sistemare il codice, in modo da renderlo presentabile e comprensibile